Project acronym: DustBot

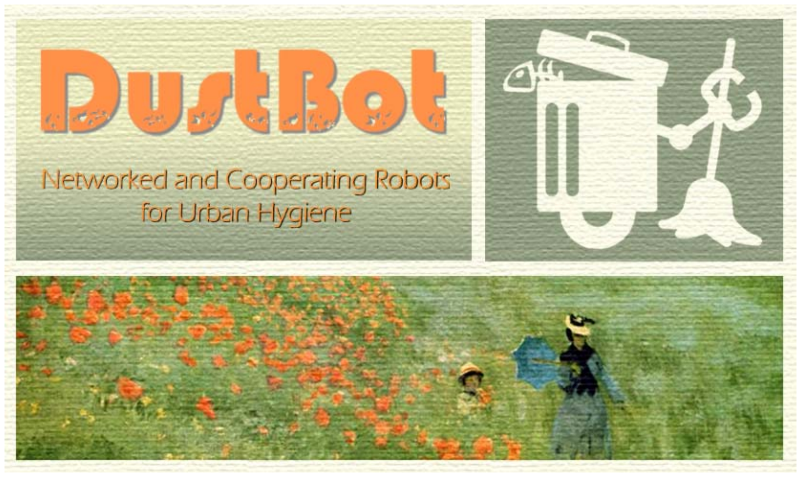
Project full title: Networked and Cooperating Robots for Urban Hygiene

Contract number: FP6-045299

Instrument Type: Specific Targeted Research or Innovation Projects

Priority: Information Society Technologies

Starting date: December 1st 2006 **Ending Date:** November 30th 2009



Information Society Technologies

Deliverable Number: D8 Additional Release

Title of Deliverable: Report on the state of the art analysis on navigation and obstacle avoidance methodologies

Task / WP related to the Deliverable: WP 4

Due Date: -

Actual Submission Date: October 31st, 2007

Organisation name of lead contractor for this deliverable: ROBOTIKER

Other contributing partners: ORU, SSSA

Revision n° 1.6

Dissemination level: PU

(PU = Public; PP = Restricted to other programme participants; RE = Restricted to a group specified by the consortium; CO = Confidential, only for members of the consortium)

Table of contents

1 Executive Summary

WorkPackage 4 (WP4) deals with the tasks related to robot localization and navigation, avoiding the potential obstacles existing on the streets and public areas. This document is the first deliverable issued within WP4, two months after the start of activities in this workpackage. Its aim is to offer a clear view of the available devices, methodologies and algorithms related to obstacle detection and avoidance, and robot localization and navigation. This deliverable D8 "*Report on the state of the art analysis on navigation and obstacle avoidance methodologies*" includes the actual state of art regarding the technologies, methods and algorithms applied for those purposes.

Chapter 2 is focused on obstacle avoidance. This task will be carried out at two levels within the DustBot controller. The high-level controller performs global obstacle avoidance in the sense that it plans obstacle-free paths based on the available representation of the environment. The low-level controller uses the current sensory information from the obstacle avoidance sensory system and performs reactive navigation, which is crucial in a non-static environment where the map cannot represent all the obstacles. The chapter includes a description of the available devices and technologies for obstacle detection (different types of cameras, infrared sensors, microwave radar, laser scanner, ultrasonic sensor) and the results obtained from first tests carried out.

A complete list of methodologies and algorithms for obstacle avoidance are also explained. Their challenge is to use real-time data from the obstacle detection hardware, calculate a safe trajectory to avoid collision with the obstacle and assure to reach the globally defined goal position.

First, some algorithms for local path planning are explained: Artificial Potential Field, Virtual Force Field (VFF), Fuzzy Controller, Vector Field Histogram (VFH). They do not attempt to find an optimal path, a task that has to be performed by a global path planner. Furthermore, the robot may get trapped in dead-end situations. In order to solve that problem, improved algorithms have been proposed and are also described in this chapter: VFH+ (it still is a local planner, but with better trajectory generation), VFH* (it adopts a look-ahead verification by projecting the trajectory of the robot several steps ahead and evaluating the consequences), Traversability Field Histogram (TFH, applied to a Segway platform). Other methods take into account the dynamic and cinematic features of the robot (Dynamic Window Approach, Curvature Velocity Space, Beam Curvature). Other types of algorithms are represented by the Nearness Diagram navigation methods (ND and ND+).

Chapter 3 is devoted to robot localization and navigation. An accurate measurement of absolute position attitude and velocity of robots is mandatory for implementing correct robot movements and path planning tasks. When attempting to determine the absolute location, the choice is usually done within three major techniques: triangulation (using the geometric properties of the triangle to compute object locations), proximity (measuring the nearness to a known set of points), scene analysis (uses the features of a scene observed from a particular vantage point to extract the location). Location system implementations generally use one or more of these techniques to locate robots, objects and people.

The main characteristic to take into account analyzing the different typologies of localization systems are the physical position (difference between the numerical and the descriptive representation), absolute or relative position, accuracy and precision, location emission, area covered, cost and limitations.

Several types of localization methods are explained and discussed: GPS (Global Positioning System based on satellite signals), Active badge (based on infrared technology, improved later on with Active Bat), Cricket (using ultrasound technology), RADAR (measuring the signal strength and signal-to-noise ratio of signals that wireless devices send), electromagnetic sensing, computer vision using 3D cameras for stereovision, pressure sensors embedded on floor, and location measurement by means of objects cooperating with other nearby objects (by sharing sensor data to factor out overall measurement).

Finally, a section dedicated to navigation algorithms covers a list of existing methods. In general, in path planning there are two situations: sometimes the mobile robot has to go from one point to a goal, while in other cases it has to cover or sweep all the area. Different conventional approaches have been developed to solve path planning problems, such as cell decomposition, road map and potential field. Most of these approaches are based on the configuration space concept. Conventional approaches are not suitable for dynamic environments because they utilize a sequential search algorithm to generate a single solution. This solution may become infeasible when a change in the environment is detected and a new solution has to be generated from scratch. To overcome the weakness of these approaches, researchers have been trying to apply other techniques to solve this problem. This situation has led to search new purely random planning methods to get a solution faster. This section of the document describes the Genetic Algorithm, different versions of Random Planning algorithms (RRT, Rapidly Exploring Random Trees), D* Algorithm, Gradient method, real-Time Visual Behaviour, sensor based navigation, the Boustrophedon cellular Decomposition and the Minimal Sum of Altitudes (MSA) Decomposition.

2 Obstacle detection and avoidance

In order to operate in outdoor environments the robot must be equipped with a high performance sensor system to gather the required information. This system must be non-contact, and must meet the requirements concerning the authorized standards (mainly the safety standard EN50100-1), in order for the robot to be sold with conformity to the Machinery Directive.

Currently several types of sensors (laser scanner and ultrasound, for example) meet these requirements and are acceptable for mobile robots. The obstacle avoidance sensory system should be technically diverse and redundant.

Obstacle avoidance will be carried out at two levels within the DustBot controller. The high-level controller performs global obstacle avoidance in the sense that this instance plans obstacle-free paths based on the available representation of the environment. The low-level controller uses the current sensory information from the obstacle avoidance sensory system and performs reactive navigation, which is crucial in a non-static environment where the map cannot represent all the obstacles.

While the high-level controller typically needs to react to asynchronous events (the arrival of a path or coverage request or an update of the map representation), the low-level controller needs constant updates upon the arrival of new sensor readings and gives control commands to the robot to avoid dynamic obstacles.

Obstacle avoidance in DustBot should also deal with constraints typical of outdoor environments, such as pavements, steps, slopes or descents; their presence will be detected by using the sensory systems. These obstacles may not be represented in the maps, and can be static or changing over the time. It is very important to place the sensors correctly in the robot. The obstacles must be detected to avoid the impact with them. This is the reason why non-contact sensors will be used.

All the perimeter of the DustBot will be sensorised, and redundancy in the information is necessary. It is important to avoid missing to detect obstacles. False detections are of less importance although care has to be taken to prevent the robot from stopping periodically to avoid contact with non-existing obstacles.

The exact location to place the sensors depends on the type of sensors and their range of detection. Some sensors will be oriented to the floor, downwards, in order to detect slopes, steps or descents and other ones will be oriented forwards to detect other static objects in general or obstacles in movement, like pedestrians and cyclist, for example.

2.1 Technologies for obstacle detection

The obstacle avoidance system will be able to detect obstacles and to know the fundamental characteristics of them like, for example, their dimensions or whether they are moving, so the detection system might identify, if possible, the type of object. In order to detect objects in a non-contact way, sensors of different technologies may be used.

2.1.1 Vision sensors

The most common are cameras using CCD arrays. The images taken with the CCD cameras will be compared to the ideal stage (without obstacles) of the maps. In this way obstacles can be detected and if pattern recognition techniques are used these obstacles can be identified. These techniques are very complex and have a high computational cost, particularly in the case of a moving camera.

The use of several cameras and stereo vision techniques gives information about the position the object relative to the DustBot.

Vision sensors are sensitive to light changes which may cause problems for mobile platforms. DustBot is a mobile platform, so complementary information is necessary. This information will be given by a system of video cameras located along the square and street for robot localization and surveillance of the environment status like it is explained in the proposal document (page 5).

2.1.2 Active infrared sensors

They can detect objects according to the angle of the infrared wave echo received. Some of them only detect objects that are nearer than 30cm, others are working for higher distances and are even able to determine the position of the object. Nevertheless, the ranges of detection hardly exceed one meter, and this fact limits the distance for the actuation of the robot in order to rectify the trajectory, mainly if the object is in movement too.

2.1.3 Passive infrared sensors

Thermal infrared sensors detect objects according to the heat they emit. They are useful for detecting pedestrians with pattern recognition techniques, though compared to vision sensors they tend to have lower resolution. They also are sensitive to high frequency intensity changes, and offer correct detections and a low number of false detections.

The problem is their high cost and that they cannot measure distances. They only detect motion and its direction.

2.1.4 Microwaves radar sensors

They are used to detect objects in short distances. Several sensors are used to measure distances using triangulation techniques. With the analysis of the wave that is reflected in the object some information about it can be obtained. The power spectral density of the reflected wave gives information about the dimensions and the dynamics of the obstacle (smaller dimensions involve narrower peaks). The reflectivity depends on the type of material. The time to reflection gives the distance to the object, the frequency of the reflected wave gives the velocity and the difference in phase among the sent wave and the received one gives the angle with respect to the object.

2.1.5 Laser scanners (or laser range finders)

They can detect objects as far as 40 m with an accuracy of only ±5 cm in a 180º field. They are fast in response, accurate, comparatively reliable in the presence of smoke or dust and insensitive to light changes. They detect objects with at least a 5% of reflectivity (almost everything). With these sensors information about the shape of the object, the distance to the object and its relative velocity can be obtained. They allow to obtain an echo from an object with remission value (the remission value depends on texture, colour and reflectivity), which helps to identify the object. The most commonly used laser scanners nowadays are 2D sensors. A severe drawback of 2D range scanners is that they are “blind” for obstacles that do not appear in the perceived plane. Therefore, 3D laser scanners are more and more used in mobile robotics. Nowadays these 3D laser range scanners consist of a 2D laser range scanner mounted on a pan/tilt unit, which typically means that the robot has to be stopped in order to make a 3D scan. However, laser range scanners that measure in more than just a single plane are available or are currently being developed (example: SICK scanner with several fixed planes).

2.1.6 Time-of-flight cameras

An alternative to 3D laser scanners is provided by time-of-flight (TOF) cameras. This sensor is based on CMOS technology and offers depth measurements for each pixel. TOF cameras are active sensors in that the scene is illuminated with modulated infrared light. Depth measurements are obtained (without scanning) by measuring the phase shift in each pixel (Ringbeck et al., 2007). Current TOF cameras such as the Photon Mixer Device (PMD) provide intensity and depth measurements with a maximal resolution of 160×120 pixels (19k PMD) up to an unambiguous range of 7.5 m at a frequency of 15-30 Hz. The nominal depth resolution is specified as 6 mm.

TOF cameras have the potential to become a standard sensor for mobile robots due to the large amount of 3D data they can deliver in real-time and because they are based on relatively inexpensive CMOS technology. The currently rather poor resolution can be increased by using data fusion with a standard CMOS camera (Andreasson et al., 2006; Jenke et al., 2007). Since they rely on active illumination, TOF cameras face problems with bright sunlight (with sophisticated background light suppression techniques, outdoor operation is possible to some degree, however) and the sensor technology is not yet fully mature.

2.1.7 Ultrasonic range sensors (sonar)

They use the same operation principle as laser scanners. They are cheaper, but less accurate. They also may present some drawbacks: potential interferences with the environment or with the multiple sensors of the robot (crosstalk), false responses due to specular reflections or porous surfaces of the obstacle, dependence on temperature and humidity, comparatively small range capacity due to a signal loss of approximately 0.06 dB/m for a 20 KHz signal (double loss if the reflected wave is measured), and low angular resolution because of the difficulty to focus a sound beam. This could be a drawback if detailed 3D modelling of the obstacles was necessary, but it is not the case for the application considered here.

2.2 Test on sensors and results

As a low cost and powerful solution for object detection several active infrared sensors as well as ultrasonic sensors have been evaluated. Their range and accuracy are enough for the DustBot requirements.

Regarding the infrared sensors, both digital as well as analogue sensors with different distance ranges have been considered.

For test purposes, 3 digital sensors for short range object detection have been tested:

- Sharp GP2D15 (5-25 cm)
- Sharp GP2Y0D340K (5-40 cm)
- Sharp GP2Y0D02YK (5-80 cm)

In order to verify whether the sensors are insensitive to changing object's materials and surface textures, several experiments have been performed with clothes worn by humans, a square wooden pole (10 x 10 cm), a metal cylinder (steel), a stone and a plastic trash can.

The results of these experiments are represented in Fig. 1 – Fig. 3.

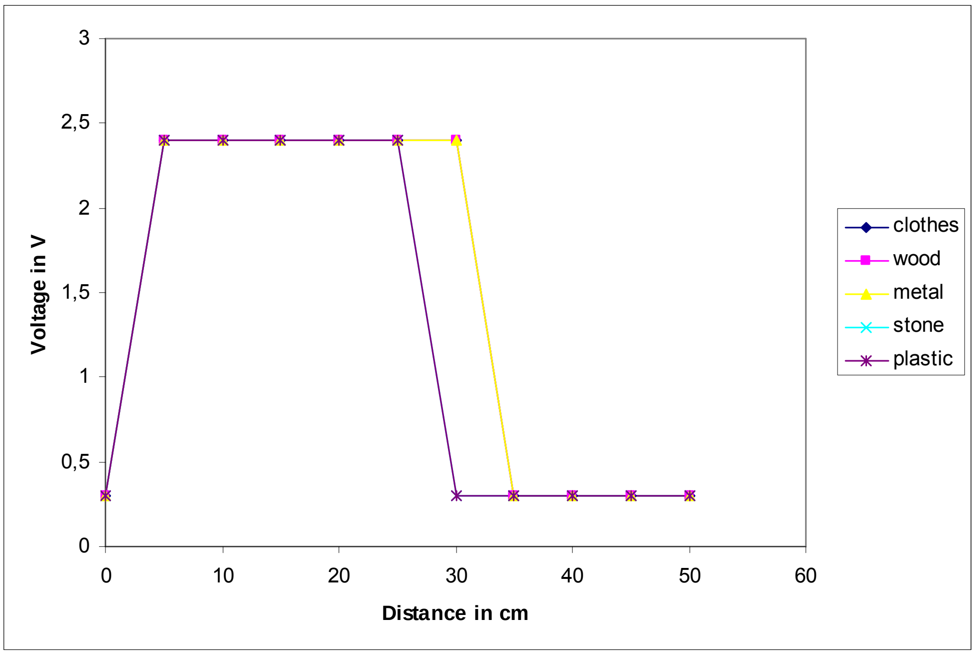


Fig. 1: Distance Characteristics for Sharp GP2D15

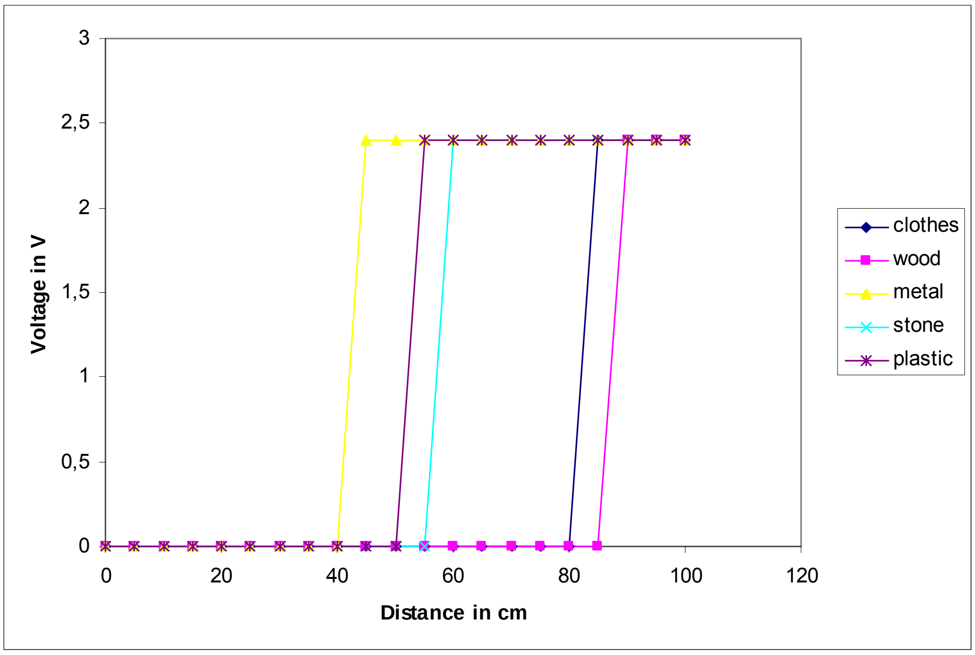


Fig. 2: Distance Characteristics for Sharp GP2Y0D340K

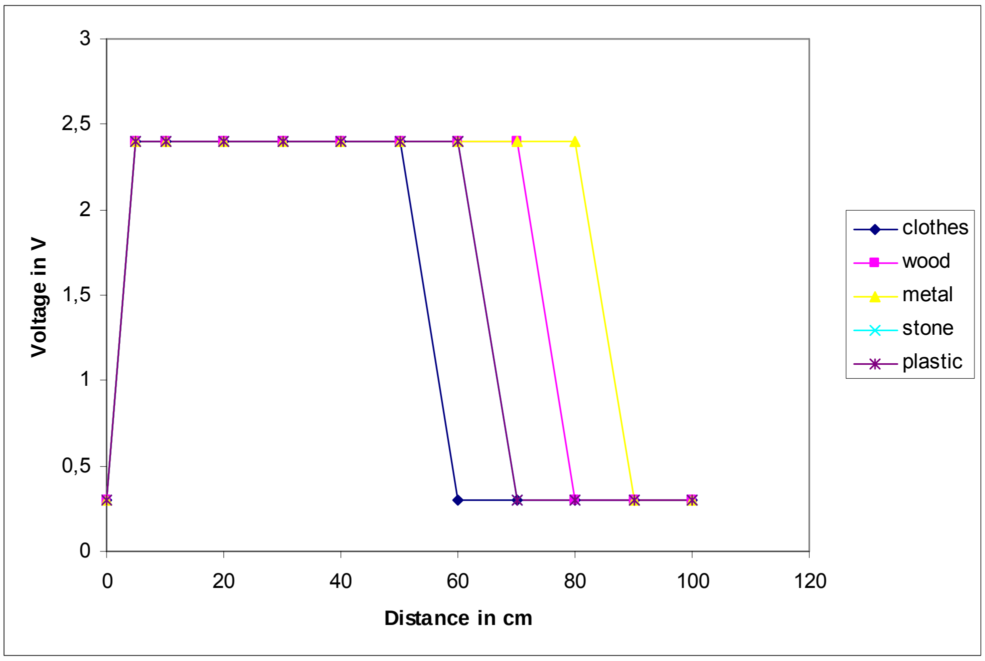


Fig.3: Distance Characteristics for Sharp GP2Y0D02YK

In general, good agreement with the data given in the data sheets of the sensors can be observed. The 25 cm sensor reliably detected all kind of objects within a distance of at most 25 cm. The 50 cm sensor detected the steel cylinder, which was the object with the most reflective surface, at a distance of 45 cm whereas other objects were detected even at farther distances. The 80 cm sensor as well showed the desired results for the metal cylinder, which was correctly detected at a distance of 80 cm. Other objects with a less reflecting surface could only be detected when they approached nearer to the sensor (up to 50 cm for textile structure of humans' clothes).

In order to be able to detect objects that are farther away and to be able to estimate their real distance, analogue sensors have been tested as well.

Two different types, one for short range detection and one for long range detection have been considered:

- Sharp GP2Y0A02YK (20-150 cm)
- Sharp GP2Y0A700K0F (100-550 cm)

Again, experiments with different kind of materials have been performed to evaluate the differences due to changing surface textures (see Fig. 4 and Fig. 5).

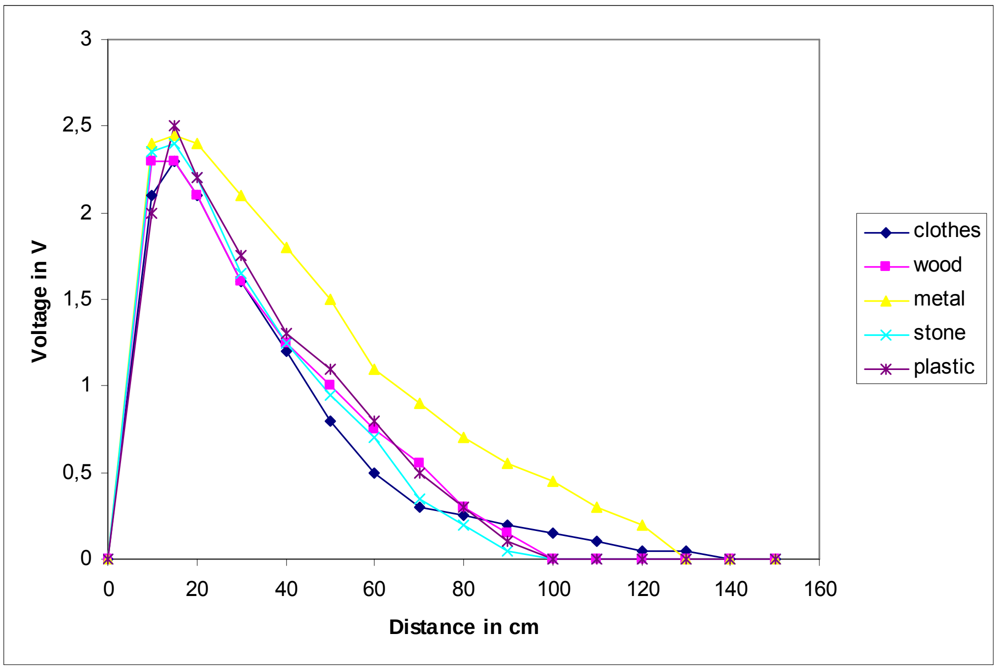


Fig. 4: Distance Characteristics for Sharp GP2Y0A02Y

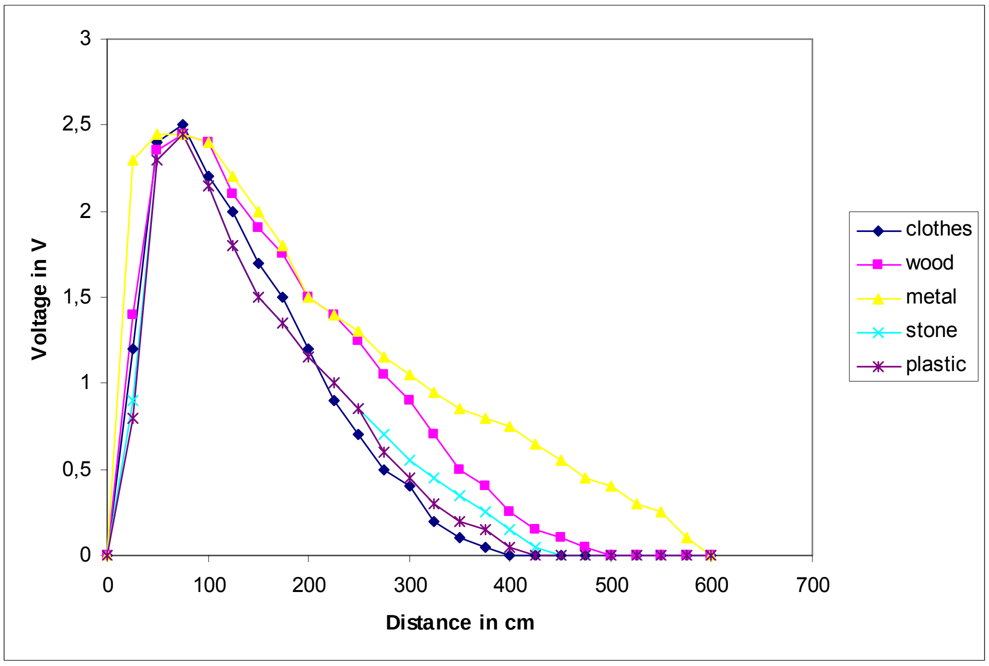


Fig. 5: Distance Characteristics for Sharp GP2Y0A700K0F

The general behaviour fits quite well to the characteristics given in the data sheet though some discrepancies especially for non metallic objects have been determined. The 150 cm sensor is able to detect all kinds of objects up to distances of about 1m. Textile surface of human clothes and the steel cylinder were even detected at a distance of 130 cm. Due to different response for different materials it is not possible to determine the exact distance of the detected object.

A similar behaviour was observed for the 550 cm sensor. Again qualitatively similar results were obtained for different materials though quantitative differences were observed. All kind of objects could be detected up to a distance of at least 4m whereas the strongly reflecting steel cylinder was even detectable up to a distance of 6m.

Furthermore, experiments were conducted to evaluate the angular deviance from the centreline, to check whether it is acceptable for correct detection. It was observed that the detectable field is rather a small cylinder than a cone. In order to be able to detect objects they have to be straight in front of the sensor (in its centreline) with a possible lateral deviation of few centimetres (1-3 cm) in horizontal as well as in vertical direction. Objects that are out of this narrow beam will not be detected. In order to avoid the use of a high number of sensors, a solution could be the mounting of one (or various) sensors on a servo for scanning a wide range in front of the robot. A combination of several fixed sensors (e.g. at the side area of the robot where intrinsic scanning is performed) and sweeping sensors will lead to better results.

As a low cost alternative to infrared sensors ultra sonic sensors have been tested. The performance of the ultra sonic ranger SRF05 (S320111) has been evaluated in a similar manner as above to check for differences for various materials and geometries.

The distance measurement of up to 4m given in the data sheet could well be verified for all tested objects and surface textures as long as the objects were placed perpendicular to the sensors centreline (see Fig. 6).

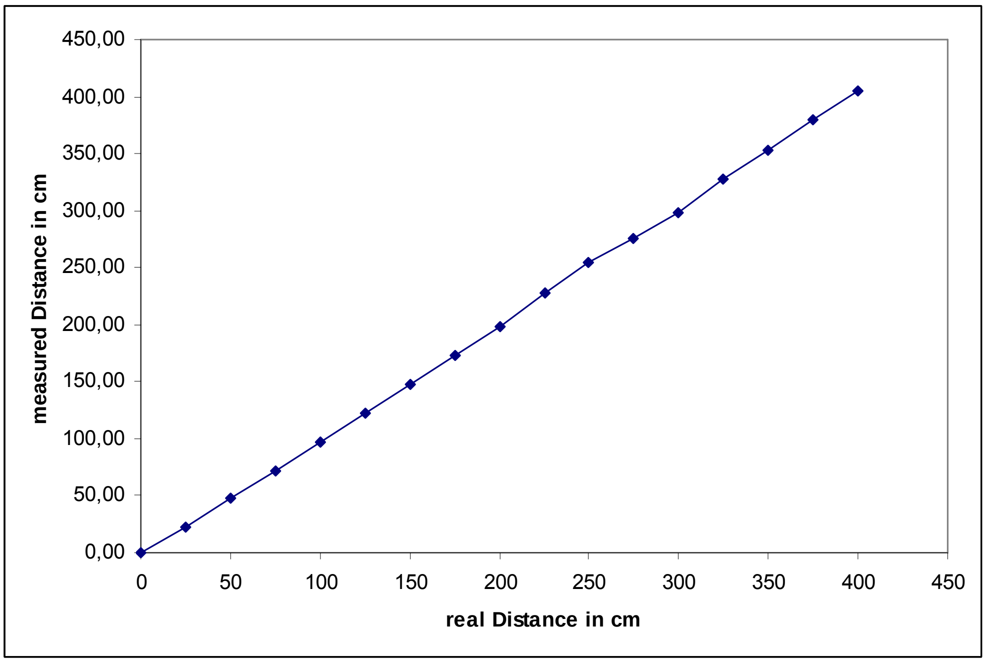


Fig.6.: Measured characteristics for SRF05

In the corresponding data sheet it is stated that the beam pattern is conical and that it is not possible to reduce or change the beam width (see Fig. 7).

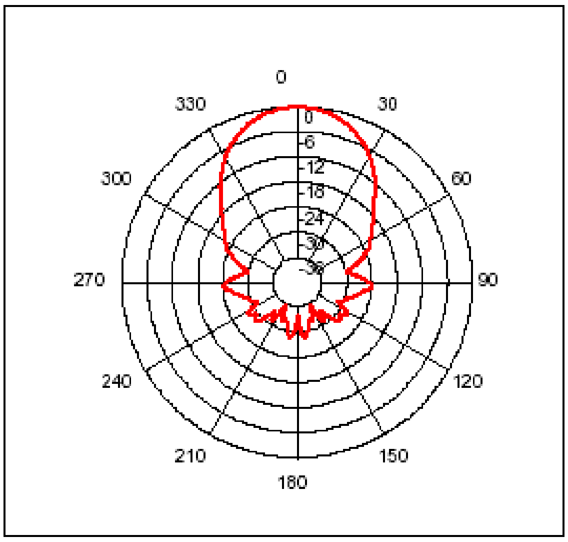


Fig. 7: Beam pattern and beam width (obtained from SRF05 data sheet)

In several experiments the real behaviour of the sensor was tested and differences for varying object dimensions were detected. A rather thin wooden pole

(10 cm) could only be detected inside a cone with an opening angle of 6° whereas the big frontal area of a plastic trash can reflect more easily the ultrasound, so it is detectable in a range of about 40° (both measurements performed at a distance of 2m – see Fig. 8). It has to be stated though that objects out of the centreline will appear to be further away from the sensor than they actually are. The reason for this behaviour is that the reflection of the ultrasound wave is weaker which makes the sensor "see" the object farther away than it really is.

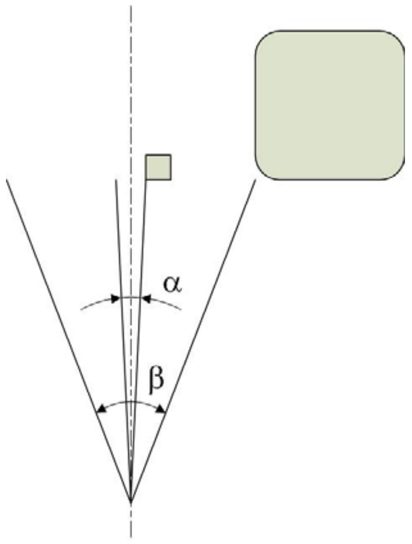


Fig. 8: Detectable range for objects of different size (α=6° for a 10x10 cm pole and β=40° for a trash can)

In comparison to the tested infrared sensors, less sensitivity to angular deviation and thus a wider detection range can be confirmed. Nonetheless, difficulties with objects with a small frontal area and especially with thin objects will arise, if they are not exactly in front of the sensor (in its centreline).

A further problem of ultrasonic sensors is that objects which are not placed perpendicular to the sensor's centreline are difficult to detect because the ultrasonic wave is not reflected back to the sensor. A 10x10 cm pole could not be detected when it was turned 45° even if it was located precisely in the centreline, whereas its detection was not a problem for perpendicular placement (measurements again performed at a distance of 2m – see Fig. 9).

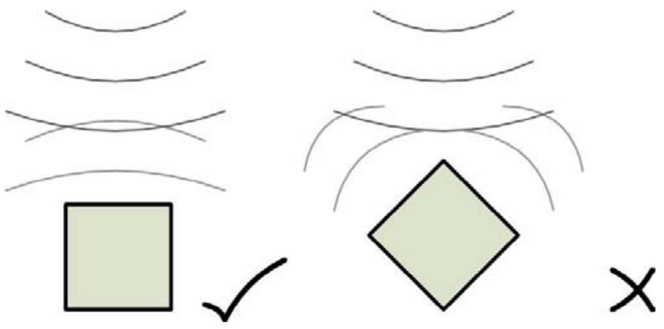


Fig. 9: Correct detection of a 10x10 cm pole for perpendicular placement, and no detection for the same pole turned by 45°

To evaluate the angular sensitivity of the sensor, experiments with a large planar surface were performed as well. For an angular deviation of $\alpha >= 20°$ it was impossible to correctly detect the distance of the plane (measurements again performed at a distance of 2m – see Fig. 10).

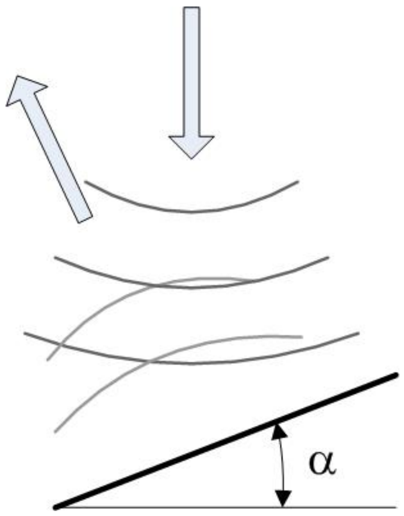


Fig. 10: Failure of distance measurement at an angular deviation of α=20°.

In order to evaluate the behaviour of commercial ultra sonic sensors, experiments with sensors integrated in the bumper of a commercial standard vehicle were performed. As for the former experiments, different materials and geometries were used and the distance of detection was measured. Slight differences could be observed but as for the tests with the SRF05 ultra sonic ranger, they were mainly caused by varying shapes rather than by varying materials. Reliable detection was observed for all kind of obstacles between 100cm for the wooden pole and 150cm for the stone.

2.3 Methodologies for obstacle avoidance

Obstacle avoidance is one of the most important tasks in mobile robotics. The challenge for obstacle avoidance algorithms is to use real-time data from the obstacle detection hardware (sensors, cameras, laser range finders,…), calculate a safe trajectory to avoid collision with the obstacle and assure to reach the globally defined goal position. In the case of a cleaning robot this is even more complicated as it is not only about eluding obstacles but also about performing autonomous coverage tasks.

In general, the here presented methods are local path planners that are based on the principle of reactive navigation. The robot has to react in real-time to the signals read by the sensors. This is necessary when autonomously moving a robot in a partly unknown environment.

2.3.1 Artificial Potential Field approach

Already in the 80's first works have been published concerning real-time obstacle avoidance. The wall following method was amongst the first that was presented as possibility for obstacle avoidance. Due to the inherent limitations of this method, soon thereafter more general methods like the artificial potential field approach (Khatib, 1985) were presented. The idea of this approach is that obstacles exert a virtual repulsive force to push away the robot from obstacles and a virtual attractive force to guide the robot to the goal position.

2.3.2 Virtual Force Field method (VFF)

Combining the method of Khatib with the concept of Probabilistic Occupancy Grid maps developed at Carnegie-Mellon University (Moravec and Elfes, 1985), Borenstein presented the Virtual Force Field (VFF) method in 1989 (Borenstein and Koren, 1989). In comparison to edge detection methods these approaches can work well with uncertain information as they respond to clusters of high likelihood for the existence of an obstacle which results in an increased robustness of the algorithm in the presence of false readings. Furthermore, updating the grid-map with sensor information and using the grid-map for navigation are two independent tasks that are performed asynchronously. Another advantage is their computational efficiency which together with the other properties made virtual force field methods very popular.

Problematic is the fact that in their basic implementation these algorithms will get stuck in local minima which are caused by U-shaped obstacles. Borenstein has presented various heuristic rules to recover from different trap situations. One recovery algorithm is the wall following method (WFM) which guides the robot out of local minimum traps by following the obstacle wall until the way to the goal position is free again. By applying this algorithm it is possible to avoid most trap situations.

2.3.3 Fuzzy Controller for Obstacle Avoidance

A completely different approach for the robot control was presented by Uribe and Urzelai (Uribe and Urzelai, 1998) who propose the use of a fuzzy controller for obstacle avoidance. Depending on the current perception of the robot's sensors the controller derives the variables for the vehicle's orientation and acceleration. The obtained trajectories are smooth and the results are satisfactory after tuning the membership functions. Nevertheless, the behaviour of the robot is similar to artificial potential field methods what means that it may easily get trapped by local minima. Only a combination with wall following algorithms can avoid that the robot remains stuck in this kind of situations.

2.3.4 Vector Field Histogram method (VFH)

Analyzing the reasons for the shortcomings of the VFF method, Borenstein and Koren came up with an improved algorithm denominated Vector Field Histogram (VFH) (Borenstein and Koren, 1991). The main problem of the VFF method lies within the fact that all repulsive forces from different obstacles are resumed to one force vector. A simple example for a malfunction of the algorithm is a robot passing through a doorway. In that case the robot could be detained from entering the doorway as repulsive forces resulting from both sides of the doorway push the robot away.

In difference to the VFF method the VFH method uses a two-stage data reduction technique. In the first reduction-step the perception of the robot's sensors updates the two-dimensional Cartesian histogram grid similar to the VFF method. In the next step, a one-dimensional polar histogram is constructed around the robot's momentary location. As in the VFF method, only an active window centred at the current robot position is considered for the calculation of the polar histogram.

The values for the drive and steer controllers are now calculated by the current goal direction and the histogram value in this direction. If the histogram value is below a prescribed threshold, the path to the goal is free and the robot moves in this direction. In case of exceeding this threshold, a valley is searched in the polar histogram close to the current goal direction and the robot is moved in this direction thus eluding a collision with the obstacle that impedes straight motion.

In case of narrow valleys the algorithm is able to decide whether the robot will fit through this narrow passage by choosing a centred path through the valley.

Nevertheless, it has to be stated, that the VFH method is a local path planner as well as earlier presented algorithms. It does not attempt to find an optimal path, a task that has to be performed by a global path planner. Furthermore, the robot may again get trapped in dead-end situations. Several heuristic rules can be applied to resolve these problems even though the resulting path will still not be optimal.

2.3.5 VFH+ method

A major improvement of the VFH method has been presented in 1998 by Ulrich and Borenstein (Ulrich and Borenstein, 1998). A first feature of the so called VFH+ method is the use of a theoretically determined low-pass filter to compensate for the width of the robot. Second, instead of using one threshold value for the calculation of the polar histogram, two threshold values are used in the VFH+ method. This improves the movement of the robot in environments with several narrow openings and avoids bringing the robot close to an obstacle. Maybe the most interesting feature of the VFH+ algorithm is the fact that it takes into account the dynamics and the kinematics of the robot. This is achieved by assuming that the possible trajectories of a mobile robot are based on circular arcs and straight lines. The radius of the arcs mostly will be directly dependant on the current speed of the robot. Thus, for a given speed of the robot it can be checked which directions may be dynamically blocked by obstacles. Algorithmically this is achieved by employing additional steps to the data reduction process. In particular, the information of dynamically blocked directions is added to the polar histogram obtaining a so-called masked polar histogram. All these features together lead to smoother robot trajectories and greater reliability.

2.3.6 VFH* method

In 2000 another improvement of the VFH+ algorithm has been presented by Ulrich and Borenstein (Ulrich and Borenstein, 2000). Whereas VFH+ is a purely local obstacle avoidance algorithm, the VFH* algorithm adopts a look-ahead verification by projecting the trajectory of the robot several steps ahead and evaluating the consequences. Although the algorithm is not purely local anymore and an A* search algorithm has to be executed, it performs still sufficiently fast to come up with the requirements for a real-time algorithm. The general idea of the algorithm is based on the VFH+ algorithm but instead of evaluating only the cost function for the current free directions, the cost function is calculated looking various steps ahead summing up the costs of the single branches.

2.3.7 Traversability Field Histogram (TFH)

Interesting for the Dustbot project is a work presented by Ye and Borenstein describing Obstacle Avoidance for a Segway Robotic Mobility Platform (Ye and Borenstein, 2004). This robotic platform is foreseen to be employed in the task of picking up bin cans from citizens. The main sensor in the system is a SICK 2D laser rangefinder which is mounted on the front end of the Segway RMP and looks forward and downward at an angle of -10° from the horizon. Thus, it is possible to scan the environment in a distance of 5 meters and the recorded data are registered in a 2-dimensional array. Obstacles or surface imperfections are represented with a value of their height in the corresponding element of the array. This information is used by the Terrain Traversability analysis TTA module which creates a Traversability Map. According to the traversability indices (TI) the local path planner generates steering and velocity commands to avoid cells with high TIs. The local path planner bases on the VFH concept but is extended by the information provided within the Traversability Map. Thus, it enhances the capability of the VFH approach from obstacle avoidance on flat ground to obstacle navigation on non-flat ground. The resulting algorithm is denominated Traversability Field Histogram.

2.3.8 Dynamic Window Approach

An approach was presented by Fox (Fox et al, 1997) taking into account the dynamic and kinematic constraints of the robot similar to the VFH+ algorithm. The method is called dynamic window approach and considers only admissible velocities which can be reached within the next time interval and which allow the robot to stop safely. The combination of translational and rotational velocity within the dynamic window is then chosen by maximizing an objective function.

2.3.9 Curvature Velocity Space method

Another method using a related approach is the curvature velocity space method (CVM) presented by Simmons (Simmons, 1996). This method chooses a point in the linear-angular velocity space which satisfies some constraints and maximizes an objective function. This objective function tries to move the robot close to the commanded direction at the highest feasible speed, while travelling the trajectory with the largest clearance from obstacles. One problem of this approach is the fact that in certain cases the largest free trajectory leads away from the target position by not finding openings in the way to this position.

2.3.10 Beam Curvature method

Fernández et al proposed an improvement for the curvature velocity space method with a radial directional method (beam method) (Fernández et al, 2004). This beam curvature method (BCM) holds the advantage that it not only maximizes the objective function but also takes into account the local direction for a collision-free space by radial projection. For that reason, BCM will find openings faster and more reliable than CVM which results in an increment in the response time to the robot. Furthermore, it will create smoother trajectories and in general reach the goal location faster than CVM.

2.3.11 Nearness diagram Navigation

The nearness diagram (ND) navigation approach is a different method that was presented by Minguez and Montano (Minguez and Montano, 2004). Similar to the VFH method a polar histogram is considered to derive actions to be taken for the robot. In difference to former methods a binary decision tree is used to derive the currently best suited action in dependence of the sensory information. Several criteria define the robots current situation. This set of situations is made up distinguishing whether the robot is in a low or high safety area, whether obstacles are only on one or on both sides of the free driving area, whether the goal location lies within the free driving area or whether the free driving area is narrow or wide. This representation is unambiguous and complete and hence fulfils the situated-activity paradigm of behavioural design (Arkin, 1999). According to each situation, an action is defined to solve the reactive navigation task. Although the actual realization differs in some points from the formerly presented VFH+ and VFH* algorithms, the final performance seems to be quite similar. The ND approach avoids local trap situations and is able to traverse narrow corridors without oscillation. One advantage of this approach is that smooth behaviour in cluttered environment is achieved without the necessity of tuning parameters. The divide and conquer strategy based on navigational situations (position of the robot, obstacles and goal position) significantly simplifies navigation thus enabling this technique to deal with more complex navigation scenarios. Nonetheless, like other local obstacle avoidance methods, this approach can not exclude the possibility to get caught in trap situations when the obstacle is not completely visible.

The ND+ method (Minguez, Osuna and Montano, 2004) improves the previous ND method with new navigational situations and a new design of the motion laws (to have motion continuity in the most common transitions between situations). Another advantage of the ND+ method is that works at more than 1000 Hz, thus the reaction to the evolution of the scenario is very rapid and it can be used when required without imposing a significant time penalty.

2.4 Preliminary conclusions

Not a unique type of sensor is able to detect and identify all kind of obstacles in any situation. That is the reason why in DustBot system MULTISENSOR solution is recommended. Different solutions are proposed:

- SEVERAL ULTRASONIC SENSORS: They can be placed in the perimeter of the robot, oriented downwards, to the floor to detect slopes, descents or steps and at a height depending on their range of detection. The number of sensors to be used will depend on the dimension of the robot and on the required field of detection.

- OTHER SENSORS for detecting moving obstacles, like pedestrians and cyclists and objects in general: They can be oriented forwards and different kinds of combined solutions may be proposed:
    - ✓ Vision sensors (+ PMD cameras) + infrared sensors.
    - ✓ Ultrasonic sensors + infrared sensors.
    - ✓ Microwaves radar sensors + vision sensors.
    - ✓ Laser scanners + vision sensors.
    - ✓ Laser scanners + microwaves radar sensors + vision sensors.

In order to decide the number and exact types of sensors, the following data should be fixed. They mainly depend on dimensions and movement characteristics of the DustBot:

- Time to respond.
- Output time.
- Range.
- Accuracy.
- Repeatability.
- Resolution.
- Sensibility.
- Linearity.

3 Methodology and algorithms for robot navigation

Navigation of Mobile Robots covers a large spectrum of different technologies and applications. It draws on some very ancient techniques, as well as some of the most advanced space science and engineering.

"Mobile Robot Navigation" covers a large spectrum of different systems, requirements and solutions.

The physical scale of a device's navigation requirements can be measured by the accuracy to which the mobile robot needs to navigate - this is the resolution of navigation. These requirements vary greatly with applications; however a first order approximation of the accuracy required can be taken from the dimensions of the vehicle itself. Any autonomous device must be able to determine its position to a resolution within at least its own dimensions, in order to be able to navigate and interact with its environment correctly.

At the small end of the scale there are robots just a few centimetres in size, which will require high precision navigation over a small range (due to energy supply constraints), while operating in a relatively tame environment. At the other end of the scale there are Jumbo jet aircraft and ocean going liners, each with some sort of auto-pilot navigation, which requires accuracy to a couple of meters (or tens of meters), over a huge (i.e. global) range, in somewhat more rugged conditions.

Three terms are used in order to help in categorizing this scale of requirements:

- *Global navigation*, which is the ability to determine one's position in absolute or map-referenced terms, and to move to a desired destination point.
- *Local navigation*, the ability to determine one's position relative to objects (stationary or moving) in the environment, and to interact with them correctly.
- *Personal navigation*, which involves being aware of the positioning of the various parts that make up oneself, in relation to each other and in handling objects.

With the jet auto-pilot example Global navigation is the major requirement, for cruising between continents. Local navigation becomes necessary where the aircraft is expected to fly autonomously in crowded airways or on approach to the runway on landing. As for the DustBot project, personal navigation is not an issue, as the vehicle is, fundamentally, one object, and should (hopefully) never come into contact with any other significant objects while under autonomous control.

The "micro" robot on the other hand, is almost exclusively interested in Personal and Local navigation. Such devices are rarely concerned with their position globally, on any traditional geographic scale. Instead their requirements are far more task based - they are concerned with their immediate environment, in particular relative to any objects relevant in the successful completion of their task. This involves Personal navigation, when it is in contact with other objects, and Local navigation for actual movement.

In general, the main focus of the scales of navigation is as follows,

- *Global*: getting between end locations.
- *Local*: carrying out a task while at a location.
- *Personal*: monitoring of the individual robot and anything in contact with it.

3.1 Technologies for robot localization

Localization is a key component in many successful autonomous robot systems (Thrun, 2001). An accurate measurement of absolute position attitude and velocity of robots is mandatory for implementing correct robot movements and path planning tasks. By some authors the robot localization problem has been stated as the most fundamental problem to providing robots truly autonomous capabilities (Cox, 1991).

The automatic localization has been addressed over the years using several methods. Many systems have been developed aiming at solving slightly different problems. These systems own very different features: they are based on different physical phenomena for estimating the robot's position; they present different power requirements and different resolution in time and space.

The general localization task presents a number of increasingly difficult problem instances (Thrun, 2001). In the position tracking problem the robot starts from a known location. The objective is to keep track of the position of the robot during the navigation through the environment. Techniques addressing this problem are called tracking or local techniques (Fox, 1999). If the robot acquires relative measurements we talk of dead reckoning: this technique has been used for a long time, ever since people started travelling around. The position estimates are based on last estimated positions and on measured speed and direction of travel: this cause an error in determining the robot's location increasing over time. In robotics, dead-reckoning is usually performed by odometry or using inertial sensors ((Borenstein, 1996).The problem of determining the position of a robot when the initial position is not known is referred to as the wake-up robot (also called "knidnapped robot") or global positioning problem. This is a more difficult problem than dead-reckoning, because the robot has to localize itself not relying on the knowledge of its starting location. Methods addressing this problem are called global techniques (Fox, 1999).

To determine the location the choice usually falls within three major methods: triangulation, proximity and scene analysis.

When attempting to determine the location, the choice is usually done within three major techniques: triangulation, proximity, scene analysis.

• The triangulation technique uses the geometric properties of the triangle to compute basing on some measurements the robot's location. Triangulation can be done using multiple distance measurements between known points (lateration), or using measures of angles or bearing relative to points with known separation (angulation).

• The proximity location sensing technique measures the nearness to a known set of points, by using a physical phenomenon with limited range. Three general approaches to this technique are: detection of the physical contact, monitoring of the wireless cellular access points and observation of ID system (such as RFID).

• The scene analysis location technique uses the features of a scene observed from a particular vantage point to extract the location of the observer of the object in the scene.

Location system implementations generally use one or more of these techniques to locate robots, objects and people. More details on these techniques are available in literature (Hightower, 2001a).

Discussing and classifying localisation system implementations many issues arise. These issues do not generally depend on the technologies or techniques a system uses. The main characteristic to take into account analyzing the different typologies of localization systems are the following:

- Physical position or symbolic location; it is the difference between the numerical or metric representation (i.e. GPS) and the descriptive or semantic representation (i.e. in the kitchen);
- Absolute or relative position (as exposed above);
- Accuracy and precision;
- Localized Location Computation; some systems require the located object to periodically broadcast, respond with, or otherwise emit telemetry to allow the external infrastructure to locate it;
- Scale; it is the area covered by the infrastructure that localizes the robot and the number of robot that can be localized simultaneously;
- Cost;
- Limitations.

A summary of the main localization technologies (both commercial and research product) are reported in Table 1 (from Hightower, 2001b); in this table the most important characteristics of such technologies are analyzed and compared. A detailed overview of the considered technologies is also reported below.

3.1.1 GPS

The Global Positioning System is maybe the most diffusely used localization system. GPS provides an excellent lateration framework for determining geographic positions relying on signals coming from the GPS satellites. The worldwide satellite constellation has reliable and ubiquitous coverage and, assuming a differential reference or using the Wide Area Augmentation System, allows receivers to compute their location with a precision of 1-5 meters (http:// www.garmin.com/aboutGPS/). GPS is currently used in aircrafts, search-and-rescue teams, hikers and navigation tools for cars.

However, GPS has some notable drawbacks. First, it does not function indoors or underground. It also functions poorly whenever there are high buildings, mountains or trees in the way blocking the signals from the satellites. Second, GPS is not accurate enough for small scale navigation. Although it is accurate down to about a meter under good circumstances, the resolution is typically too coarse to assist a robot when turning or avoiding obstacles, for example. Of course, a GPS cannot detect obstacles, which means that the robot has to have additional sensors anyway. Thus, although GPS does give you a fairly good position fix under the right circumstances, and should certainly be used to support navigation when available, it is by itself insufficient.

3.1.2 Active badge

The Active Badge location system (the first indoor badge sensing system) was developed at Olivetti Research Laboratory (now AT&T Cambridge) (Want, 1992) and is a cellular proximity system relying on diffuse infrared technology. Each agent system uses a small infrared badge. The badge emits a globally unique identifier every 10 seconds or on demand. These data are collected from fixed infrared sensors around the building. The Active Badge system computes the absolute locations of the agent systems. Active Badge system present problems when fluorescent lighting or direct sunlight are present because of the spurious infrared emissions these light sources produce. The effective range of this system is several meters. This limits cell sizes to small or medium-sized rooms. Multiple infrared beacons can be used in larger rooms.

3.1.3 Active Bat

Recently, AT&T lab has developed a location system called the Active Bat. This system uses an ultrasound time-of-flight lateration technique to reduce the problems Active Badges has (Harter, 1999). The system is based on Active Bat tags carried by the users. The controller sends a request via short-range radio; a Bat answers emitting an ultrasonic pulse to a grid of receivers fixed to the ceiling. The controller sends together with the radio frequency request packet also a synchronized reset signal to the sensors on the ceiling. The signal is sent using a wired serial network.

The time interval between the reset and the arrival of ultrasonic pulse is measured by each ceiling sensor. In this way, the ceiling sensor can compute its distance from the Bat. The local controller then sends the measured distance to a central controller, which performs the lateration. Statistical elaborations are carried out in order to eliminate erroneous sensor measurements caused by a ceiling sensor hearing a reflected ultrasound pulse not coming from the direct path from the Bat to the sensor.

3.1.4 Cricket

Another system complementing the Active Bat system (Priyantha, 2000) based on ultrasound emitters is the Cricket Location Support System. The infrastructure is created by the ultrasound emitters and the receivers that are embedded in the objects that have to be localized. In this approach the objects perform all their own triangulation computations. Cricket uses the radio frequency signal for synchronization of the time measurement and to limit the time period during which the receiver takes into consideration the sound it receives. Ultrasound pulses sensed after the end of the radio frequency packet are recognized as reflections and are discarded. An algorithm allows the use of multiple uncoordinated beacons in the same space.

3.1.5 RADAR

A system based on the IEEE 802.11 WLAN networking technology has been developed by a Microsoft Research group. This system is named RADAR (Bahl,2000) and it measures, at the base station, the strength and signal-to-noise ratio of the signal sent by the wireless device. Basing on these data it computes the 2D positions within a building. Two different versions exist: one using lateration and the other using scene analysis.

3.1.6 MotionStar magnetic Tracker

The use of electromagnetic fields to measure the position of one object is common in products for virtual reality and motion capture for computer animation. A classic position tracking method is that proposed in (Raab, 1979). Ascension offers several motion-capture solutions, such as the MotionStar DC magnetic tracker (Technical Description of DC Magnetic Trackers, 2001). These systems generate axial DC magnetic-field pulses from a transmitting antenna in a fixed position. The system measures the response to the transmitted pulse in three orthogonal axes, then, combining it with the constant effect of the earth’s magnetic field it computes the position and orientation of the object. Other technologies have been used in virtual environments and for computer animation. In (Bible, 1995) a CDMA radio ranging approach has been proposed. Many optical, infrared and mechanical motion-capture systems are produced by different companies. These systems, like Motion-Star, are used in controlled environments and are not conceived and designed to be scalable for large environments.

3.1.7 Easy Living

The use of vision technology to localize an agent, or to localize objects in an environment, has been investigated by several groups of researchers. One result is the Microsoft Research's Easy Living. This system employs the Digicolps real-time stereo cameras to localize objects in a home environment (Krumm, 2000). The vision-based systems typically use large amounts of processing power to analyze the different frames captured by the vision system hardware. Multi-modal processing (silhouette, skin colour and face pattern) have been proved (Darrell, 1998) to significantly enhance accuracy in localization.

3.1.8 Smart Floor

In Georgia Tech's Smart Floor system embedded pressure sensors (Orr, 2000) capture footfalls, and the system uses the data for position tracking. With this system the moving agent to be localized does not have to carry a device. The problem with the Smart Floor, however, is the poor scalability because the floor of each building has to be embedded with the pressure sensors.

3.1.9 Ad hoc location sensing

Ad hoc location sensing idea is to succeed in localizing objects without using an infrastructure or central control. All the entities become mobile objects owning the same sensing capabilities. Estimates of the positions are produced from the relative distances measured between the entities. These are relative measurements that can be absolute ones if some objects occupy known locations. For computing the distances triangulation, scene analysis or proximity can be used. One locating system based on this idea is the SpotON system (Hightower, 2000). It uses lateration using low-cost tags. The tags use radio signal attenuation to estimate the distance from one tag to another. The accuracy of the estimate is improved correlating multiple measurements.

Technology	Technique	Absolute	Relative	Accuracy and precision available	Scale	Cost	Limitations
GPS	Radio time-of-flight lateration	X		1-5 meters (95-99%)	24 satellites worldwide	Expensive infrastructure $100 receivers	Not indoors
Active Badges	Diffuse infrared cellular proximity	X		Room size	1 base per room, badge per base per 10 sec	Administration costs, cheap tags and bases	Sunlight and fluorescent light interfere with infrared
Active Bats	Ultrasound time-of-flight lateration	X		9 cm (95%)	1 base per 10 square meters, 25 computations per room per sec	Administration costs, cheap tags and sensors	Required ceiling sensor grid
MotionStar	Scene analysis lateration	X		1 mm, 1 ms, 0.1° (nearly 100 %)	Controller per scene, 108 sensors per scene	Controlled scenes, expensive hardware	Control unit tether, precise installation
Cricket	Proximity, lateration	x	x	4 × 4 ft. regions (≈ 100 %)	≈ 1 beacon per 16 square ft.	$10 beacons and receivers	No central management receiver computation
MSR RADAR	802.11 RF scene analysis and triangulation	X		3-4.3 m (50 %)	3 bases per floor	802.11 network installation, ≈ $100 wireless NICs	Wireless NICs required
Easy Living	Vision, triangulation	X		Variable	3 cameras per small room	Processing power, installation cameras	Ubiquitous public cameras
Smart Floor	Physical contact proximity	X		Spacing of pressure sensors (100 %)	Complete sensor grid per floor	Installation of sensor grid, creation of football training dataset	Recognition may not scale to large populations
SpotON	Ad hoc lateration		X	Depends on cluster size	Cluster at least 2 tags	$30 per tag, no infrastructure	Attenuation less accurate than time-of-flight

Table 1 – Summary of main location sensing technologies.

3.2 Algorithms for robot navigation

In Robot Navigation, on the theoretical side, a robot is faced with a number of algorithmic issues that are geometric in nature. This includes mapping a given environment, searching all possible locations in such an environment, or localizing the robot's position on a given map; typically, available information is visibility-based, but motion-planning may also require the computation of a collision-free trajectory for a rigid body, if one exists. These geometric aspects are pursued in the field of **Computational Geometry**, where a lot of expertise has been developed, including deep results on visibility problems and motion planning.

Another crucial feature of robot navigation is that path-planning has to be performed without full knowledge of all necessary data; such information only becomes available during the course of the robot's motion, requiring optimization with incomplete information. Complete knowledge of the scenario only becomes known after a strategy has actually been applied. This means that an algorithm has to protect against various possibilities (including faulty sensors or inaccurate data), instead of basing its decisions on a complete description of the given information.

Motion planning is one of the important tasks in intelligent control of an autonomous mobile robot.

In a generic way, the problem of motion planning lies in carrying the robot from one initial point to a final one in a space free of collisions. This problem has largely been studied (Latombe, 1991; Laumont et al., 1994; Muñoz, 1995), and there is a great number of effective methods to solve the planning problem in real time, such as potential fields, visibility graphs, Voronoi diagrams, etc. However, in complex environments, there are more degrees of freedom, the robot presents more kinematic restrictions, and the limitations of the methods and the necessary time to find a solution are greater, too.

Path planning algorithms are classified according to completeness as exact and heuristic (Hwang, 1992). Exact algorithms aim to find an optimal solution if one exists, or prove that there is no feasible solution. On the other hand, heuristic algorithms aim to search for a good quality solution in a short time. Exact algorithms are usually computationally expensive and assume perfect knowledge about the environment; however, heuristic algorithms may fail to find a solution for difficult problems.

Different conventional approaches have been developed to solve path planning problems (Latombe, 1991) such as cell decomposition, road map and potential field. Most of these approaches are based on the configuration space concept (Lozano-Pérez, 1979). Conventional approaches are not suitable for dynamic environments because they utilize a sequential search algorithm to generate a single solution. This solution may become infeasible when a change in the environment is detected and a new solution has to be generated from scratch. To overcome the weakness of these approaches, researchers have been trying to apply other techniques to solve this problem.

One known planning method is the potential fields method, which although it is fast, has a problem with the local minima. Random generation algorithms are used to obtain local planning towards a new minimum (Latombe, 1991).

This situation has led to search new purely random planning methods to get a solution faster. Among the most widespread is the "Rapidly Exploring Random Trees" (LaValle, 1998), called RRT. There are new versions of this method in order to improve it, like "RRT- Ext-Con" (Kuffner y LaValle, 2000; LaValle y Kuffner, 2001) and "ERRT" (Bruce y Veloso, 2002).

In general, in path planning there are two situations: sometimes the mobile robot has to go from one point to a goal, while in other cases it has to cover or sweep all the area. These situations are very different, so the path-planning requires different solutions in each case.

In addition, sometimes the area to cover is known and mapped in advance, but in other situations the environment is unknown. In fact, the tasks of planning trajectories for a mobile robot have received considerable attention in the research literature. Most of the works assume that the robot has a complete and accurate model of its environment before it begins to move; less attention has been paid to the problem of partially known environments.

There are many researches and studies to solve these problems. Different algorithms exist to solve different and particular situations. Some of them are explained in following paragraphs, grouped into two types of problems: "Point to Goal (PtG)" navigation and "Coverage (C)" navigation.

3.2.1 Global and Local Path Planning

The path-planning problem is usually defined as follows (Sugihara, 1997): "Given a robot and a description of an environment, plan a path between two specific locations. The path must be collision-free (feasible) and satisfy certain optimization criteria." The path-planning component is divided in two sections: global path planning and local path planning.

Global path planning requires the environment to be completely known and the terrain should be static. In this approach the algorithm generates a complete path from the start point to the destination point before the robot starts its motion. Local path planning is done while the robot is moving and involve the algorithm capable of producing a new path in response to environmental changes. Assuming that there are no obstacles in the navigation area, the shortest path between the start point and the end point is a straight line. The robot will proceed along this path until an obstacle is detected. At this point, this path-planning algorithm is utilized to find a feasible path around the obstacle. After avoiding the obstacle, the robot continues to navigate toward the end-point along a straight line (in this system the robot moves in a vertical or horizontal direction, not diagonally; hence, it will try to approximate a straight line) until (1) the robot detects another obstacle or (2) the desired position is reached. An example of local path planning is shown in Fig. 22.

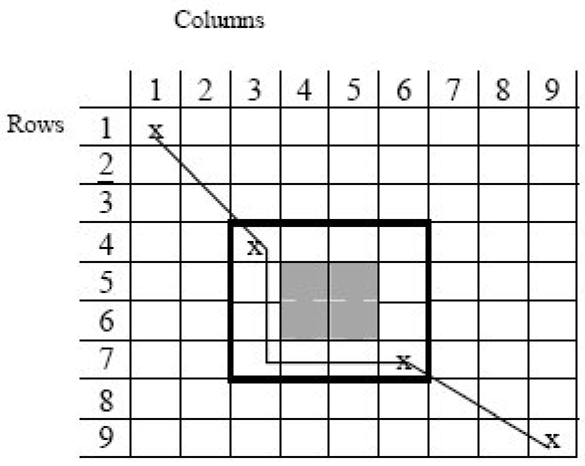


Figure 22. Path-planning example for local obstacle avoidance, applied on a subsection of the search space.

3.2.2 Genetic Algorithm (PtG)

Robot path planning is part of a larger class of problems pertaining to scheduling and routing, and is known to be NP-hard (NP-complete) (Obitko, 1998). Thus, a heuristic optimization approach is recommended as shown by Hwang (Hwang, 1992). One of these approaches is the use of genetic algorithms. A genetic algorithm (GA) is an evolutionary problem solving method, where the solution to a problem evolves after a number of iterations. A proposed solution with the GA method to the path-planning problem is the best feasible path among the pool of all possible solutions.

There have been several contemporary applications of genetic algorithms to the robot navigation problem. One approach is to combine fuzzy logic with genetic algorithms (Arsene, 1999; Kubota, 1999; Pratihar, 1999). In this approach, the genotype structure represents fuzzy rules that guide the robot navigation, so the genetic algorithm evolves the best set of rules. While this approach can produce a feasible path through an uncertain environment, the genotype structure becomes very complex, as it needs to represent a variety of fuzzy rules. A complex genotype structure can take a long time to process in a genetic algorithm, which affects the realtime performance of the robot during navigation.

Another approach is to use genotype structures that represent local distance and direction, as opposed to representing an entire path (Cazangi, 2002; Di Gesu, 2000; Gallardo, 1998; Vadakkepat, 2000). While these are simple to process and allow for faster real-time performance, the local viewpoint of these methods may not allow the robot to reach its target. Some methods have relatively simple genotype structures that can represent feasible paths, but require complex decoders and fitness functions (Hocaoglu, 2001; Sugihara, 1997; Xiao, 1997). This can also affect real-time response.

Simplifying the models used to represent navigation paths will reduce the processing time of the genetic algorithm. Research has focused on improving the genetic algorithm performance by simplifying the genotype structure.

Genetic algorithms are a class of adaptive methods that can be used to solve search and optimization problems involving large search spaces.

The following are general specifications for a GA-based local path-planning approach:

- A map of the room in which the path planning takes place is known. The path planner will determine the length and the width of the search space and then apply a grid system to the room, similar to a chessboard. Thus, the room is divided into rows and columns. The locations of known obstacles are marked as “occupied cells” in the grid.
- The row and column coordinates of the start-point and the end-point of the desired robot’s movement are also known.
- The robot is allowed to move on all “free” cells, where the centre of the robot moves along an imaginary line from the centre of one cell to the centre of another cell.

There are two types of robot movements:

- *Row-Wise* Movement: In a row-based movement, the robot starts moving row by row from the start-point to the end-point. In other words, any horizontal line in the search space will meet the path only once. Therefore, in this movement, the robot always has to go forward and it does not have the capability of going back (up) to the previous row.
- *Column-Wise* Movement: In a column-based movement, the robot will start moving toward its destination column by column to the right. In other words, any vertical line in the search space will meet the path only once. Therefore, in this movement, the robot always has to move from left to right, and it does not have the capability of moving back to the left.

The structure must have sufficient information about the entire path from the start point to the end-point in order to be able to represent it. There are two variables defined: *Path-Localization* and *Path-Direction*, and this technique only allows row-wise movements.

A new instruction flag is defined for each path, called *Path-Flag*. This *Flag* instructs the next movement type for each step of the movement, allowing the robot to plan either a row-wise or a column-wise movement according to the search space arrangements, to combine both row-wise and column-wise paths while planning for a single path. This causes the robot to fail for complex environments that require the planning problem adds then a new variable: *Path-Switch*. The path has now more flexibility to switch between the two movement modes.

Gene Structure: *Path-Flag*: a 1-bit flag for each chromosome. The main responsibility of this bit is to tell the robot whether the next step of the movement is row-wise or column-wise. During the entire robot movement, the decoder will check this instruction bit before each movement step. The next movement type will be based on the information provided by this flag.

Gene Structure: *Path-Location*: if the robot is required to go row-wise (Path-Flag = 0), a gene's position within a chromosome corresponds to a row-number (y-coordinate). The value, stored in a gene, in a variable called path-location, corresponds to a column number (x-coordinate).

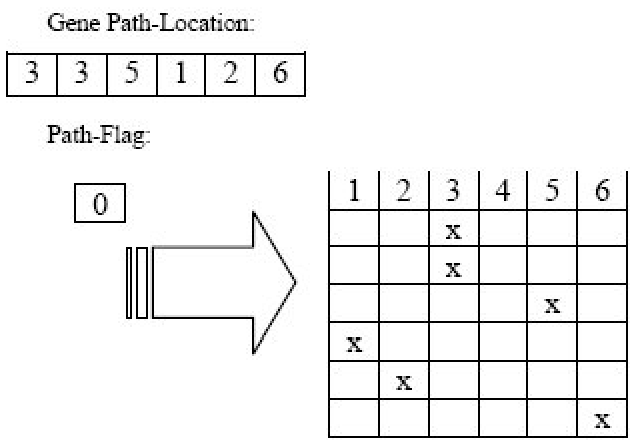


Figure 23. Example of Path-Location for row-wise movement

On the other hand, for the column-wise movement (Path-Flag = 1), a gene's position within a chromosome corresponds to a column-number (x-coordinate). Then, the value stored in that gene corresponds to a row-number (y- coordinate).

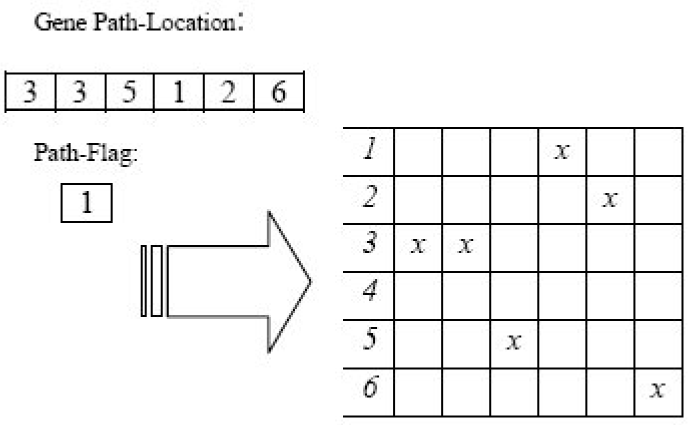


Figure 24. Example of Path-Location for column-wise movement

Gene Structure: *Path-Direction*: The gene structure described so far only represents vertices ('corner points' or 'intermediate steps') of a path. To send a robot on a straight line directly from a centre of one vertex to the centre of the next vertex would mean that the robot moves on a diagonal line across many adjacent cells. This could cause problems if not all adjacent cells (that the robot is to traverse going from one cell to the next) are free of obstacles, as shown in Figure 25. A better approach is to go to the side (horizontal) first, turn, and then go down (vertical), or vice versa. To indicate the first direction the robot will turn to proceed to the next vertex, a second variable called Path-Direction is added to the gene structure. Direction is a two-state variable (Boolean), which has either the value 1 or 0 for horizontal or vertical directions respectively. The length of the direction array is one less than the length of the location array, since there is no direction instruction for the last location.

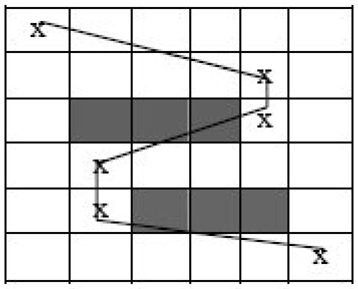


Figure 25. Problem with diagonal movement of the robot

Now the connection, and therefore the path, from one vertex to the next one is not a diagonal line, but a combination of a horizontal / vertical movement. Since the first direction that the robot turns to can be either horizontal (solid line) or vertical (dotted line), there are two possible ways to get from one vertex to the next one for each step. The introduced variable Path-direction indicates which of the two ways the robot will use to go to the next vertex.

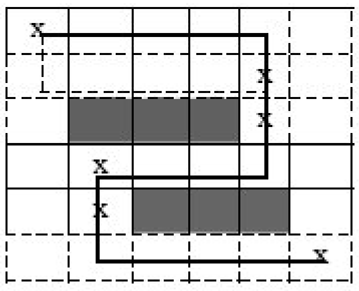


Figure 26. The path with horizontal / vertical instead of diagonal movement

It is obvious in Figure 26, one exception where the *pathdirection* variable will not affect the robot movement direction is when the two consecutive movement steps are either in the same column (for the row-wise movement) or in the same row (for column-wise movement). In either case, there is only one way to go from one vertex to the next one, which is a straight horizontal or vertical line.

Gene Structure: Path-Switch: to combine both row-wise and column-wise paths while planning for a single path. *Path-Switch* variable enables the robot to switch back and forth between a row-wise (r.w.) and a column-wise (c.w.) movement in a single path. This array contains two switching numbers. Therefore, the robot can switch a maximum of two times from row-wise to column-wise and vice versa within a search space. The values that are stored in this array are integers and are in the range of 1 to the total length or width of the environment. The numbers stored in this array indicate the location where the robot has to switch from r.w. to c.w. movement or vice versa.

The switching numbers could be any number from 1 to the total number of the search space rows or columns.

The number stored in each switching point indicates the location of the gene in which the robot has to switch. For instance, switching numbers 2,5 means the robot is switching two times: first at gene locations 2, then at location 5.

3.2.3 Random Planning (PtG)

Random Planning parts from a space of configurations divided in pieces that form a regular grid. Each square has a value of potential according to the proximity to an obstacle or the nearness to the start or goal points. The system generates a path with a gradient vector derived form the potential field. This technique allows the system to follow always the direction which minimizes the potential field value. The goal is to get the absolute minimum which will be located in the destination configuration. But the resultant field can have some local minima, though always higher than the absolute minimum.

When the system reaches a local minimum, the potential is not zero, but the gradient keeps the system in the reached configuration. To solve this inconvenience it resorts to a random generation method. Random movements are planed to let the system leave the minimum and, then, it applies again the gradient method. This process will continue until it finds a new minimum (Latombe, 1991).

With the success of the random methods the possibility to use this technique in an exclusive way has emerged, eliminating the cost of processing for the calculus of the potential. They have to be simpler, to compete in speed and to compensate the lack of potential field information in the path planning algorithm. One of these methods is called “Rapidly Exploring Random Trees”, RRT (LaValle, 1998). It does not require to compute a potential field, and consequently saves processing time. In addition, the RRT assesses an equiprobable exploration for the whole configuration space. Finally, it is simple, fast and of easy extension towards complex stages.

RRT algorithm: its original objective was to build one exploration tree to cover uniformly all the collision free space. This algorithm only has to generate a tree able to explore in an equiprobable way the free space. It’s not the most appropriate method to find the path among two points.

The adaptation of the RRT algorithm to connect one initial configuration to a final one is obtained replacing the original RRT algorithm with the *RRT- basic bidirectional* algorithm. This algorithm is based in the construction of two trees that leave from the origin and destination points at the same time.

This algorithm has some improvements. It presents one very restrictive condition, which requires that in the same iteration both trees meet in the same point. To get rid of this restriction go, it has been developed the *RRT-Ext-Con* algorithm (Kuffner and LaValle, 2000). The trees growth can be directed from one to another, instead of grow in places where the interconnection is difficult. To achieve this the *RRT-Ext-Ext* algorithm has been developed (La Valle and Kuffner, 2001).

The *RRT-Ext-Ext* algorithm makes agile the connection among the trees. With a very little variation, it is possible to attribute an ability to let each tree to drive its growth towards its homologous. This makes it be more competitive against other algorithms purely random. In this way each tree spends half of its time exploring the free space, and the other half, looking for its partner.

The *RRT-Ext-Con* algorithm, instead of adding a new segment to the tree, adds consecutive segments until it reaches the objective configuration or one obstacle (Kuffner and LaValle, 2000). It attenuates in this way the problem of the excessive restrictiveness of the *basic bidirectional* algorithm. The only disadvantage is a greater computational cost that is compensated in many stages with the greater efficiency of the algorithm. This algorithm presents a great efficiency when it plans trajectories for holonomic systems. On the opposite, like it is shown in (Cheng and LaValle, 2001), the *RRT-Ext-Ext* algorithm represents the best option when it tries to plan trajectories for non holonomic systems.

The obtained trees in the application of the different RRT algorithms can result very complex to be covered. Normally, the trees are susceptible of simplification (for example, sometimes the end of both trees could joint with a simple straight line).

Therefore, in the practical applications of the RRT method, it is convenient to apply a postprocess to the obtained trees to reduce their irregular topology. In this way, like in other planning techniques, is common to apply simplification algorithms, very fast and simple, which proceed from the information given by the planificator and generate, in an iterative way, the simplest path. Such algorithms usually have a small computational cost. The postprocessing algorithm must be designed according to the particular application, constructing one more element for the planning system.

3.2.4 D* Algorithm (PtG)

This algorithm is capable of planning paths in unknown, partially known, and changing environments in an efficient, optimal, and complete manner. The name of the algorithm, D*, was chosen because it is *dynamic* in the sense that arc cost parameters can change during the problem solving process.

It is assumed that the environment is completely known before the robot begins its traverse. The optimal algorithms search a state space (e.g., visibility graph, grid cells) using the distance transform (Jarvis, 1985) or heuristics (Nilsson, 1980) to find the lowest cost path from the robot's start state to the goal state. Cost can be defined to be distance travelled, energy expended, time exposed to danger, etc.

Unfortunately, the robot may have partial or no information about the environment before it begins its traverse but is equipped with a sensor that is capable of measuring the environment as it moves. One approach to path planning in this scenario is to generate a "global" path using the known information and then attempt to "locally" circumvent obstacles on the route detected by the sensors (Goto, 1987). If the route is completely obstructed, a new global path is planned. Lumelsky (Lumelsky, 1986) initially assumes the environment to be devoid of obstacles and moves the robot directly toward the goal. If an obstacle obstructs the path, the robot moves around the perimeter until the point on the obstacle nearest to the goal is found. The robot then proceeds to move directly towards the goal again.

It is possible to generate optimal behaviour by computing an optimal path from the known map information, moving the robot along the path until either it reaches the goal or its sensors detect a discrepancy between the map and the environment, updating the map, and then re-planning a new optimal path from the robot's current location to the goal.

There is a new algorithm for generating optimal paths for a robot, operating with a sensor and a map of the environment. The map can be complete, empty, or contain partial information about the environment. Regions of the environment can be unknown, the map may contain approximate information, stochastic models for occupancy, or even heuristic estimates. The algorithm is functionally equivalent to the brute-force, optimal replanner, but it is far more efficient.

3.2.5 The Genetic Algorithm Planner

A *Genetic Algorithm Planner (GAP)* was proposed for solving the path planning problem in static and dynamic mobile robot environments. The GAP is based on a variable-length representation. A generic fitness function is used to combine the objectives of the problem. Different evolutionary operators are applied: some are random-based, and others use problem-specific domain knowledge. Various techniques are investigated to ensure that the GAP is appropriate for dynamic environments.

In order to allow the algorithm to operate on the entire working space, vertex graphs are used to represent obstacles in the robot environment. Each obstacle is represented by an ordered list of vertices with no restrictions on the shapes or the sizes of the obstacles.

A *chromosome* represents a path as a sequence of nodes, where each node contains an “x” and “y” coordinate of a point. The first node is the starting point (or the robot’s current location) and the last node represents the destination point. The number of knot nodes (intermediate nodes) in the path is variable. By using this representation the algorithm is able to search the entire space and adapt to the new changes in the environment with no extra map adjustments. Figure 27 shows the linked list data structure used to accommodate the variable length path. The initial population is generated randomly, where each path has a random number of nodes.

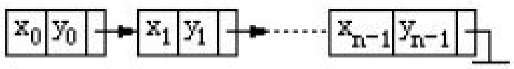


Figure 27: Chromosome data structure

A feasible path is evaluated according to the length, smoothness and clearance. A linear combination of these factors is illustrated with the following equation:

$$eval(p) = w_d.dist(p) + w_s.smooth(p) + w_c.clear(p)$$

where w_d, w_s, and w_c represent the weights on the total cost and dist(p), smooth(p) and clear(p) are defined as follows:

- $dist(p) = \sum_{i=1}^{n-1} d(s_i)$, where d(si) is the distance between two adjacent nodes.

- $smooth(p) = \sum_{i=2}^{n-1} e^{a(\theta_i - \alpha)}$, where θi is the angle between the extension of the two line segments connecting the ith knot point. α is the desired steering angle and "a" is a coefficient.
- $clear(p) = \sum_{i=1}^{n-2} e^{a(g_i - \tau)}$, where gi is the smallest distance from the i^{th} segment to all obstacles, and τ is the desired clearance distance the "a" coefficient, which is referred to as the map coefficient which is a problem dependent coefficient and is defined as follows:

$$a = MAX(\frac{A}{2 * O_A}, 2)$$

Where A is the total map area, and O_A is total obstacle area. Accordingly, the smoothness and clearance factors increase for simple environments, whereas it becomes small in a crowded environment.
Figure 28 illustrates all the components involved in the feasible path evaluation function.

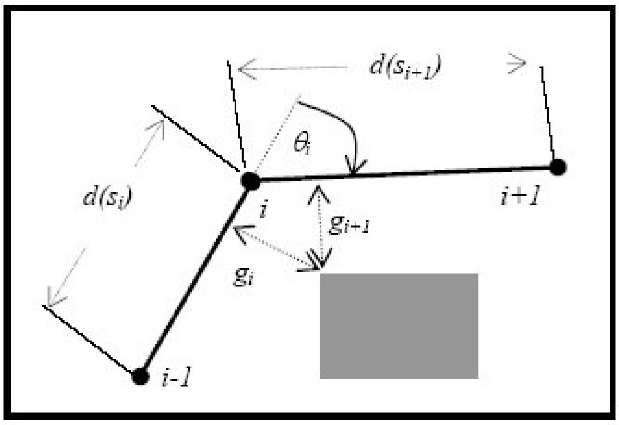


Figure 28 illustrates all the components involved in the feasible path evaluation function

An unfeasible path is evaluated according to the number of intersections with obstacles and the ratio between the numbers of feasible segments and unfeasible segments. A penalty function is used to make the least feasible path better than any unfeasible path.

3.2.6 Gradient Methods for Real Time Robot Control (PtG)

Another method for local navigation, the *gradient method*, computes optimal paths to waypoint goals. The method is efficient enough to be used for real-time control. It overcomes the limitations of other local control paradigms because it computes a complete set of optimal paths to every point in the workspace, avoiding local minima and other control problems.

Most current controllers are combinations of different techniques, using motion planning for a global path and other techniques to deal with uncertain or unknown objects. Motion planning generates an initial path based on prior knowledge of the environment, and then the path is adjusted as the robot senses obstacles that lie in the way of the path.

The *gradient method*, continuously calculates an optimal path to a waypoint goal. The concept of optimality is derived by assigning costs to a path, based on its length and closeness to obstacles, as well as any other criteria that may be chosen. The gradient method computes a navigation function in the local space of the robot, such that the gradient of the navigation function represents the direction of the lowest-cost path at every point in the space. The method is efficient enough to be computed at a 10 Hz rate with modest computational resources. The gradient method can generate the lowest-cost path in a static and completely known environment.

The *navigation function* assigns a potential field value to every point in the space. Travelling along the gradient of the navigation potential yields the minimum cost path to the goalset from any point in the space.

To find a path with minimum cost to some point in the goalset, that path is represented by an ordered set of points in the sample space,

$P=\{p_1, p_2, \dots\}$

Abbreviate a path that starts at point *k* by P_k.

The cost of a path is an arbitrary function of the (discretized) path, F(P).

Make the assumption that the path cost is separable into the sum of an *intrinsic cost* of being at a point, along with an *adjacency cost* of moving from one point to the next:

$$F(P)=\sum_i I(p_i)+\sum_i A(p_i,p_{i+1})$$

I and *A* can be arbitrary functions. *I* will represent the cost of traversing through the given point, and will be set according to the domain characteristics, e.g., a high cost will be assigned to being near an obstacle. Other possibilities are to have higher costs for unknown regions, slippery regions, etc.

The path length can be taken into account by assigning *A* to be proportional to the Euclidean distance the robot travels between the two points; then the sum of *A* gives a cost proportional to the path length.

A *navigation function* *N* is the assignment of a potential field value to every element of the configuration space, such that the goalset is always "downhill" no matter where you are in the space (Latombe, 1991). Navigation functions, unlike potential field methods in general, have the characteristic that it is impossible to get stuck in a local minimum, and no search is required to determine a direction to go to arrive at the goalset.

The key idea behind the gradient method is to assign the navigation function at a point to be the cost of the minimal cost path that starts at that point.

$$N_k = \min_{P_k} F(P_k)$$

where as before, the path P_k starts at point k.

If the intrinsic costs are zero, then the navigation function just represents the distance to the nearest goalset point. Travelling in the direction of the gradient of N yields the fastest reduction of path costs, i.e., the minimum distance to a goalset point. In the general case, travelling along the gradient is a minimum cost path to the goalset.

Navigation functions for small-dimension spaces have been computed by a *wavefront algorithm* (Akin, 1990; Thrun, 1998). The goalset points are assigned a value of 0. At each iteration, the points with value *n* are expanded to their nearest (rectilinear) neighbours, giving them a value of *n*+1 if they are not already assigned, and are not obstacles. The process repeats until all points have been assigned. It takes time of the order of the number of points in the space, which is why it can only be used in small spaces.

The wavefront algorithm never backtracks, because at each point it advances the navigation function only to those points that have a value one higher than any other assigned point.

Linear programming is a generalization of the wavefront algorithm, which is called **LPN**. The cost of **LPN** is again the order of the number of points in the space, and it reduces to the wavefront algorithm under the appropriate conditions on the navigation function.

Initially all goalset points are assigned a value 0, and every other point an infinite cost. The goalset points are put in an *active list* of points. At each iteration of the algorithm, it operates on each point on the active list, removing it from the list and updating its neighbours.

It is possible to show that the **LPN** algorithm computes the least-cost path to every point in the workspace. At each point in the workspace, the gradient of the navigation function computed by LPN, if it exists, points in the direction of a least-cost path to the goalset.

The LPN algorithm runs in time proportional to the number of points in the workspace. On a modest PC (266 MHz Pentium), a C implementation averages about 1 µs per workspace point. For a 10 m by 10 m workspace, with a grid size of 10 cm, it will take 10 ms to calculate the navigation function. Thus, the LPN algorithm is suitable for realtime control. Using a 10 Hz rate, it is fast enough to run the robot efficiently at speeds up to 1 m/s.

3.2.7 Real-Time Visual Behaviours for navigating a mobile robot (PtG)

The use of vision in mobile robotics has become wide spread. Humans have a good sense of direction, and can navigate without using precise coordinates for localisation (McCleary, 1974). In other methods, navigation is done relative to landmarks (Schone, 1984). The goal is to provide a mobile robot with sufficient visual behaviours to navigate freely in an unknown environment.

This method of using video signal produces obstacle avoidance by searching at frame rate for free space in each frame of the video signal. While the robot is travelling through the environment it could be given the purpose of finding a goal, and then using this for navigational purposes. This is done by providing the robot with goal templates (or landmarks).

The vision system continuously searches its input for the landmark template. If a landmark template is detected, the robot can execute several actions. If at any time a match is found a motion vector toward the landmark is sent to the robot. This will guide the robot towards the goal. This is a visual servoing behaviour.

In a map, a landmark can be included together with information about its location of a goal. This provides the robot with an internal representation of the goal. This is similar to the idea of McFarland *et al.* (McFarland, 1993) of goal-directed behaviour. This provides the robot with a desire to move towards a goal. The behaviour produces a motion vector that is fed into the robot's locomotion system resulting in motion towards the goal. Combining this behaviour with the goal seeking behaviour results in a behaviour which moves the robot between landmarks. The robot can navigate by using visual landmarks as cues to robot actions. The position of the landmarks can be changed without affecting the system, thus yielding a robust system.

3.2.8 Sensor-based navigation by Minguez et al. (PtG)

Up to now global navigation systems have been presented in this chapter whereas local reactive path planners have only been discussed in an isolated manner in chapter 2.3. Especially in the case of a dynamic environment which is not completely specifiable with an a priori map it is obvious that the overall problem cannot be solved by these systems individually. Thus, it makes sense to focus on synthesizing a control mode that incorporates these methodologies, and not on extending both worlds separately (Arkin, 1990).

Especially Minguez et al (Minguez, Montesano and Montano, 2004) have performed extended research in this field. Their sensor-based robot navigation system is formed by an architecture that integrates three modules with the following functionalities: model construction, motion planning and reactive navigation.

Globally the system works as follows (Figure 29): given a laser scan and the odometry of the vehicle, the model builder incorporates this information into the existing model. Next, the information of obstacles and free space in the grid is used by the planner module to compute the course to follow to reach the goal. Finally, the reactive module uses the information of the obstacles contained in the grid and information of the tactical planner to generate the motion (to drive the vehicle free of collisions towards the goal). The motion is executed by the vehicle controller and the process restarts with a new sensorial measurement.

It is important to stress that the three modules work synchronously within the perception - action cycle.

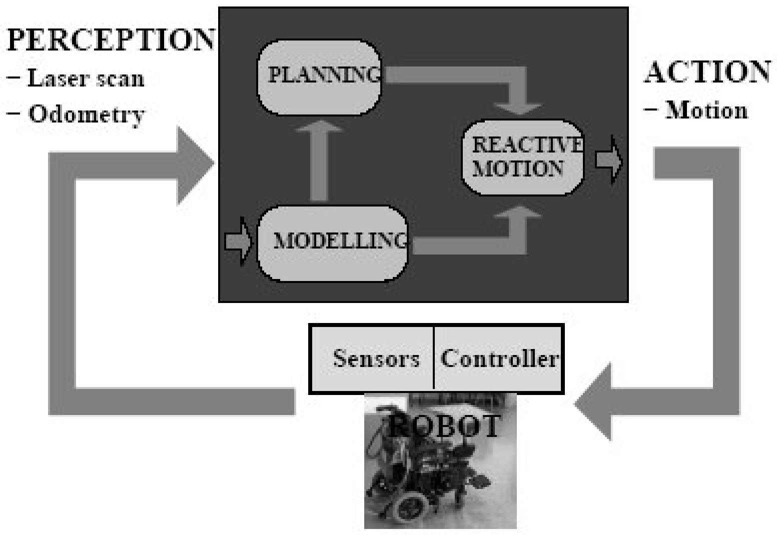


Figure 29: Overview of the sensor-based navigation system

Model Builder Module:

The function of this module is to integrate the sensorial measures to construct a model of the environment. A binary occupancy grid is chosen because it is an efficient structure from which it turns out simple to obtain the free space (the one of interest for the movement). The cells are assigned either free or occupied state by scanning the workspace with a laser range finder resulting in high precision. The grid has a fixed size that represents a limited part of the workspace (large enough to represent the portion of space necessary to solve the navigation task) and whose position is recomputed to maintain the robot in its central zone. The design and supervision of this module include three parts: (i) the use of a technique of scan matching to improve the vehicle odometry, for example using the Iterative Dual Correspondence algorithm (Lu, 1997), which searches for correspondences between two consecutive laser scans in order to estimate the rigid motion. (ii) the integration of the laser measures in the model, using the Bresenham algorithm (Foley, 1990), and (iii) the supervision of model position to maintain the robot centred.

Planning Module:

This module uses a motion planner to obtain tactical information to avoid trap situations and cyclical motions. In the course of the years several different motion planners have been implemented to conduct this task (Montesano, Minguez and Montano, 2006). In a first approach, the planner constructs a navigation function (NF1) over the grid of the previous module. Then it computes a path to the destination using a steepest descendent strategy. This navigation function is free of potential minima (if a path exists, it is found), and it can be efficiently executed in real time. In a first improvement a planner was developed by Minguez et al independently but similar to the *Gap Navigation Trees*. The idea behind this planner is to construct a graph of reachable points of the space, instead of an analytical path as many classical planners do. The planner iteratively constructs a graph whose nodes are locations in the space and the arcs are tunnels of free space that joins them. When the goal is reached, the current tunnel contains a path to the goal. The advantage of this planner is the computation time since in average it is more efficient than computing a local path from scratch with a navigation function. Following this development the application of the *D* Lite* planner was explored as well. The principle of this planner is to locally modify the previous path (available from the previous step) using only the changes in the environment. This strategy is by far more efficient than recomputing the path from scratch (up to two orders of magnitude).

Reactive Module:

As reactive module the Nearness Diagram Navigation method (ND) and subsequently the ND+ method came to use. For a closer description of this method see chapter 2.3.11.

Next sections cover the “Coverage path planning”, which is the determination of a path that a robot must take in order to pass over each point in an environment. Applications include vacuuming, floor scrubbing, and inspection.

3.2.9 The Boustrophedon Cellular Decomposition (C)

The *boustrophedon* (literally means “the way of the ox”) cellular decomposition is an exact cellular decomposition approach for the purposes of coverage. Each cell in the boustrophedon is covered with simple back and forth motions. Therefore, coverage is reduced to finding an exhaustive path through a graph which represents the adjacency relationships of the cells in the boustrophedon decomposition. *Coverage path planning* determines a path that guarantees that an agent will pass over every point in a given environment.

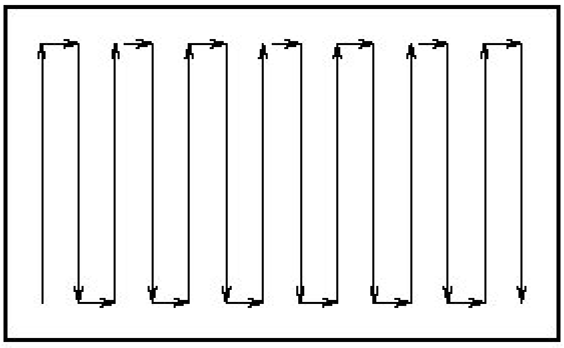


Figure 30: Boustrophedon Path

Cellular decomposition is a motion planning technique in which the free configuration space (set of all robot configurations where the robot does not overlap an obstacle) is decomposed into cells in such a way that the union of the cells is the original free space. Each cell can be represented as a node in a graph, where adjacent cells have an edge connecting their corresponding nodes. This graph is called an adjacency graph. If each cell can be covered by the robot, then the floor coverage problem reduces to determining a walk through the adjacency graph that visits each node at least once.

One popular cellular decomposition technique, which can yield a complete coverage path solution, is the trapezoidal decomposition (Latombe, 1991), also known as the slab method (Preparata, 1985) in which the robot's free space is decomposed into trapezoidal cells. Since each cell is a trapezoid, coverage in each cell can easily be achieved with simple back and forth motions (see Figure 30). Coverage of the environment is achieved by visiting each cell in the adjacency graph. The trapezoidal decomposition approach assumes that a vertical line, termed a *slice*, sweeps left to right through a bounded environment which is populated with *polygonal* obstacles. Cells are formed via a sequence of *open* and *close* operations which occur when the slice encounters an *event*, an instance in which a slice intersects a vertex of a polygon. There are three types of events: IN, OUT, and MIDDLE. Loosely speaking, at an IN event the current cell is closed (thereby completing its construction) and two new cells are opened (thereby initiating their construction). See Figure 31.

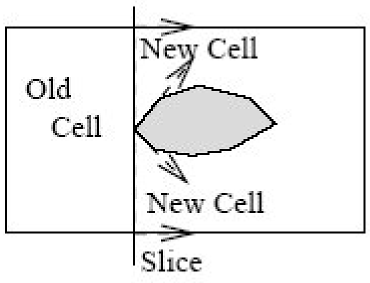


Figure 31: In Event

An OUT event is the reverse: two cells are closed, and a new one is opened. See Figure 32.

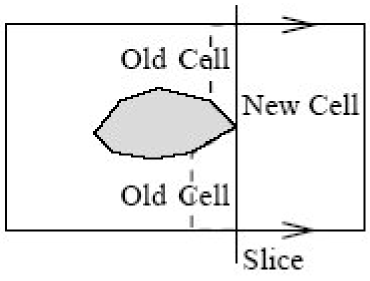


Figure 32: Out Event

The IN event can be viewed as one cell breaking up into two cells, whereas the OUT event occurs as two cells merge into one. At a MIDDLE event, the current cell is closed, and a new one is formed. The result of these operations is a free space that is broken down into trapezoidal cells.

Unfortunately, the trapezoidal approach requires too many redundant back and forth motion to guarantee completeness. Another drawback of the trapezoidal approach is that it requires the environment to be polygonal.

The boustrophedon cellular decomposition is an enhancement of the trapezoidal decomposition and is designed to minimize the number of excess lengthwise motions. In essence, all cells between IN and OUT events are merged into one cell. Figure 33 shows the trapezoidal decomposition and Figure 34 the boustrophedon decomposition. The boustrophedon decomposition has a fewer number of cells.

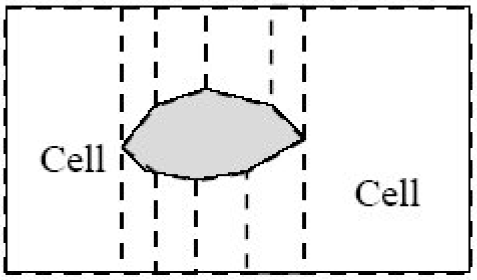


Figure 33: Trapezoidal Decomposition

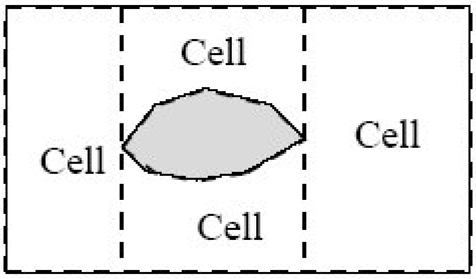


Figure 34: Boustrophedon Decomposition

In the boustrophedon cellular decomposition, like in the trapezoidal decomposition, at an IN event, where the connectivity of the slice increases, the current cell is closed and two new cells are opened (Figure 35).

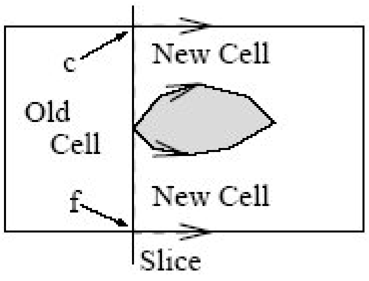


Figure 35: In Event

Conversely, at an OUT event, where the connectivity of the slice decreases, the two current cells are closed and one new cell is opened (Figure 36).

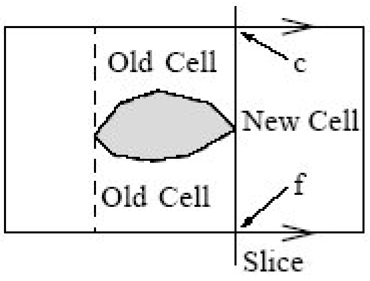


Figure 36: Out Event

The difference between the trapezoidal decomposition and boustrophedon decomposition approach is with the middle events: at the MIDDLE events, do not open nor close a cell, but rather simply update the current cell.

A depth-first-like graph search algorithm outputs a path list that represents an exhaustive walk through the adjacency graph. A walk through the path list constitutes an exhaustive walk through the adjacency graph. Finally, the actual path for the robot to take is computed using the above described path list. When the robot enters an "unvisited" cell, the boustrophedic motion is planned, and then a path to the next cell in the path list in planned. When the robot enters a "visited" cell, it simply plans a path through that cell to the next cell in the path list. These two actions are repeated until the end of the path list is reached, i.e., until each cell has been "visited".

In the boustrophedon decomposition method, the middle event is replaced with two more types of events: FLOOR and CEILING. FLOOR events correspond to vertices that are on the *top* of the polygonal obstacle and the CEILING events correspond to vertices that are on the *bottom* of the obstacle.

The input to the algorithm is a list of polygons whose vertices are listed in counter-clockwise-order.

An event is a vertex of a polygon and may be associated with additional information; it contains the location of the event, its type, and pointer(s) to the edge (or edges) that is (are) associated with it. The event structure has up to two types of pointers to edges: floor pointers and ceiling pointers. An IN event's ceiling pointer points to the next edge emanating from the event and the floor pointer points to the previous edge terminating at the event. See Figure 37.

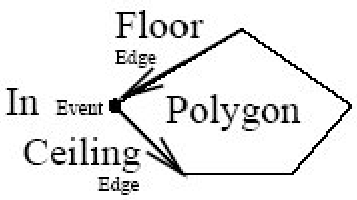


Figure 37: In Event

Conversely, the OUT event's floor pointer points to the next edge emanating from it and the ceiling pointer points to the edge terminating at the event. A CEILING event only has a ceiling pointer which points to the edge emanating from the event. A FLOOR event only has a floor pointer which points to the edge terminating at the event.

When considering a particular polygon, the algorithm first finds the IN event of the polygon. The algorithm walks through the vertex list of a polygon until it encounters the left-most vertex. This vertex, and its related information, is inserted into an event list. Since the vertices are ordered in a counter clockwise fashion the next sequence of vertices are CEILING events. Recall that although these vertices correspond to the underside of the polygon, they are CEILING events because they correspond to the ceiling of the cell that is immediately below the polygon.

The algorithm walks through the polygon list, inserting each vertex as a CEILING event, until the algorithm encounters the right-most vertex. This vertex, and its associated information, is inserted into the events list as an OUT vertex. The remaining vertices correspond to FLOOR events.

As the events are encountered, they are inserted into an order events list sorted by the x-coordinate of the event. The insertion process is O (n log n) where n is the total number of edges (or vertices) in polygonal environment.

A cell can be represented by two lists: a list of floor edges and a list of ceiling edges, both of which bound the cell. Therefore, the cell structure contains two pointers to a list of edges: a floor pointer and a ceiling pointer. The cell structure also contains a linked list of pointers to neighbouring cells. Finally, the cell structure has two flags: visited and cleaned, which will be used in the algorithm.

3.2.10 The Minimal Sum of Altitudes (MSA) Decomposition (C)

In robotics, one basic approach to the coverage problem is to decompose the region into sub regions, select a sequence of those sub regions, and then generate a path that covers each sub region in turn.

Under certain assumptions, the cost to cover a polygonal sub region is proportional to its minimum altitude. An optimal decomposition then minimizes the sum of sub region altitudes.

A coverage algorithm must generate what it is called a *coverage path*, i.e. a detailed sequence of motion commands for a robot over a specified region. An optimal coverage algorithm would return a coverage path that minimizes, for example, the time required to execute that path.

Several existing algorithms take the following basic approach to generating a coverage path: the region to be covered is decomposed into sub regions, a *Travelling Salesman* algorithm is applied to generate a sequence of sub regions to visit, and a coverage path is generated from this sequence that covers each sub region in turn.

These algorithms all use a single line sweep in order to decompose the coverage region into sub regions, and these sub regions are individually covered using a back and forth motion in rows perpendicular to the *sweep direction*. In existing algorithms, all sub regions use the same sweep direction.

After finishing one row, the robot must turn around to start the next row, minimizing the number of these turns is the most important factor in an efficient solution. The number of turns is directly related to the orientation of the sub region (measured along the sweep direction), so the optimality criterion is to minimize the sum of sub region altitudes.

By allowing different sweep directions to be assigned to each sub region of a decomposition, it can be produced a lower sum of sub region altitudes and thus a cheaper coverage path: the *minimum sum of altitudes* (MSA) decomposition. An algorithm to generate this decomposition creates an initial decomposition based on multiple line sweeps and then uses dynamic programming to group sub regions and assign sweep directions to each sub region.

Previous works exits about this decomposition:

- Choset and Pignon describe an off-line planning algorithm for polygonal worlds which explicitly performs line sweep decomposition (the “Boustrophedon” decomposition) and creates a sequence of sub regions (cells) using an heuristic Travelling Salesman algorithm. Hert et al. describe an online algorithm for non polygonal worlds which implicitly uses a line sweep decomposition and a heuristic Travelling Salesman algorithm. Schmidt and Hofner use an off-line planning algorithm to generate a coverage path based on a line sweep decomposition.
- Kurabayashi et al. (Kurabayashi, 1996) describe an off-line algorithm for planning coverage paths for multiple robots. It appears to generate a single coverage path, based on both “direction parallel” and “contour-parallel” motion. Zelinsky et al. (Zelinsky, 1994) describes a grid-based coverage algorithm.
- More recent results include: Gabriely and Rimon, who formulated a coverage algorithm based on travelling about the perimeter of a minimum spanning tree that fills the coverage region; Butler et al. (Butler, 2000), who created a distributed algorithm for multiple robots to cover an unknown rectilinear environment; and Choset et al. (Choset, 2000) who have extended the Boustrophedon decomposition to higher dimension Euclidean spaces.

A coverage algorithm must return a *coverage path*: a detailed sequence of motion commands for the robot to sweep over all points in the coverage region. Optimal coverage is a difficult problem because it is not clear what an optimal coverage path would look like; there are many qualitatively different coverage paths for a given region.

The time to cover a sub region using back and forth motion consists of the time to travel along the rows plus the time to turn around at the end of a row. As illustrated in Figure 38, covering a sub region in different directions produces rows of approximately the same total length; however, there can be a large difference in the number of turns required. Furthermore, turns take a significant amount of time - the robot must slow down, make the turn, and then accelerate. It's important therefore to minimize the number of turns required, and this is proportional to the altitude of the sub region measured along the sweep direction.

Under these assumptions, the MSA decomposition will result in optimal coverage because the total cost of covering the sub regions is the minimum for this class of solutions.

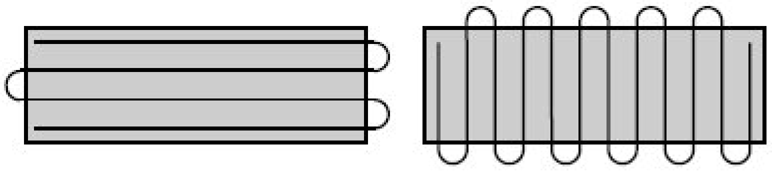


Figure 38: The number of turns is the main factor in the cost difference of covering a region along different sweep directions

This algorithm first decomposes the coverage region into cells based on multiple line sweeps and then applies dynamic programming to group the cells into sub regions and assign sweep directions to each sub region.

For each edge orientation (of the boundary, a hole, or their convex hulls), a line sweep is performed using a sweep direction perpendicular to such an edge. Each line sweep is done independently, but all decompositions are overlaid, in effect taking the dividing lines introduced by all line sweeps.

The resulting cells are monotone with respect to all sweep directions under consideration. In addition, all no convex edges are extended until they hit an obstacle boundary or the coverage region boundary.

From the initial decomposition, an adjacency graph is created (each node represents a cell, and two nodes are connected if they share an edge). This adjacency graph may have cycles, even if there are no holes in the coverage region. The graph G is the entire adjacency graph from the initial decomposition.

The basis of the dynamic programming formulation is to either split this graph into two sub graphs, thus creating two smaller sub problems, or to create one sub region from all the cells in G (and cover the entire sub region under one sweep direction). If the graph is split, two (individually) connected sub graphs G_1 and G_2 are created. These sub graphs together contain all the edges from G except those that connect a node from G_1 to a node in G_2.

The minimum sum of altitudes is:

$$S(G) = \min \left\{ C(G), \min_i S(G_1^i) + S(G_2^i) \right\}$$

where *i* iterates over all possible ways to split the graph G into two connected sub graphs and C(G) returns the cost of covering all cells corresponding to nodes in G as one sub region. When there is only one node in the graph, S(G)=C(G).

The function C(G) must consider all the sweep directions under consideration to determine the cost for covering the cells in G as a single region. For some (or possibly all) coverage directions, this sub region may not be monotone, in which case the cost is assigned, C(G) returns the minimum cost over all sweep directions under consideration.

Figure 39 shows an example of the first level of decomposing a problem; Figure 40 shows the three line sweep decompositions and the optimal solution produced by the MSA decomposition algorithm.

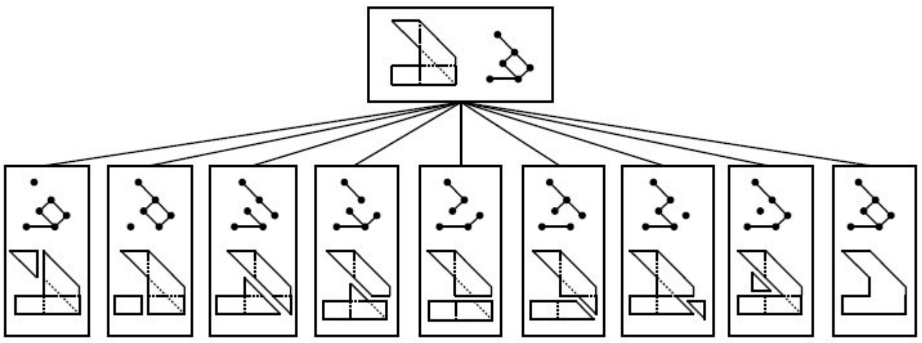


Figure 39: The top box shows a coverage region and the initial decomposition, and the corresponding adjacency graph for the proposed MSA decomposition algorithm. There are 8 ways that this graph can be decomposed into two separate connected graphs, and the corresponding division of the coverage region is shown. The rightmost choice represents covering all cells as a single region.

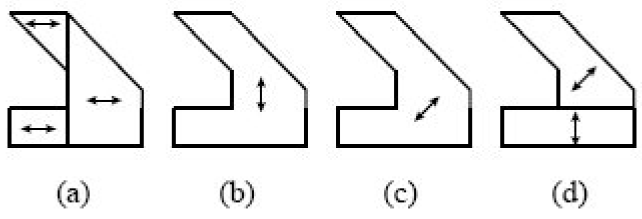


Figure 40: Figures (a) through (c) show the three line sweep decompositions of this region. Figure (d) shows an optimal MSA decomposition produced by the algorithm. The sum of sub region altitudes for Figures (a) through (d), respectively, is 10, 7, 6.4, and 5.5.

To sum up, the basic approach is the decomposing of the coverage region into sub regions, selecting a sequence of sub regions, and generating a coverage path that covers each sub region in turn.

The MSA decomposition divides a coverage region into sub regions such that the sum of sub region altitudes is minimized. Behind this criterion is the idea that sub regions are covered with back and forth motion along rows perpendicular to the sweep direction. The number of turns at the end of rows is the most important factor affected by the orientation of the sweep direction. The number of turns is directly related to the sub region altitude (with respect to the sweep direction).

An algorithm creates an initial decomposition by performing multiple line sweeps and extending no convex edges. Dynamic programming is then used to group cells in the initial decomposition into sub regions and to assign each sub region a sweep direction.

4 References

Andreasson, H., Triebel, R. and Lilienthal, A.J. “Vision Based Interpolation of 3D Laser Scans”. Proceedings of the International Conference on Autonomous Robots and Agents (ICARA), 2006.

Arkin, R. C., “Integrating behavioural, perceptual and world knowledge in reactive navigation”. Robotics and Autonomous Systems 6, pp. 105-122, 1990.

Arkin, R., “Behavior-Based Robotics”. The MIT Press, 1999.

Arsene, C.T.C. and Zalzala, A.M.S., “Control of Autonomous Robots Using Fuzzy Logic Controllers Tuned by Genetic Algorithms”, Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), pp. 428-435, 1999.

Bahl, P. and Padmanabhan, V., “RADAR: An In-Building RF-Based User Location and Tracking System” Proc. IEEE Infocom 2000, IEEE CS Press, Los Alamitos, Calif., pp. 775-784, 2000.

Bible, S.R., Zyda, M. and Brutzman, D., “Using Spread- Spectrum Ranging Techniques for Position Tracking in a Virtual Environment” Second IEEE Workshop Networked Realities, http://www.npsnet.org/~zyda/pubs/ NR95-Paper-Bible.pdf.

Borenstein, J. and Koren, Y., “Real-time Obstacle Avoidance for Fast Mobile Robots”. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, No. 5, pp. 1179-1187, 1989.

Borenstein, J. and Koren, Y., “The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots”. IEEE Journal of Robotics and Automation, Vol. 7, No. 3, pp. 278-288, 1991.

Borenstein, J. and Feng, L., “Measurement and correction of systematic odometry errors in mobile robots”, IEEE Transactions on Robotics and Automation 12, pp 869-880, 1996.

Brown, R.G, “Introduction to random signal analysis and Kalman filtering”, John Wiley and Sons, 1983.

Butler, Z., Rizzi, A., and Hollis, R., “Complete distributed coverage of rectilinear environments”. In Fourth international workshop on the algorithmic foundations of robotics, 2000.

Cazangi, R.R. and Figuieredo, M., “Simultaneous Emergence of Conflicting Basic Behaviors and Their Coordination in an Evolutionary Autonomous Navigation System”, Proc. 2002 IEEE Conf. on Evol. Comp. (CEC '02), IEEE, 2002.

Choset, H., Acar, E., Rizzi, A., and Luntz, J., “Exact cellular decompositions in terms of critical points of Morse functions”. In IEEE International Conference on Robotics and Automation, 2000.

Computational Geometry Algorithms Library (CGAL). http://www.cgal.org/

Cox, I. J., “Blanche: Position estimation for an autonomous robot vehicle”, Autonomous Mobile Robots: Control, Planning, and Architecture (Vol. 2), IEEE Computer Society Press, Los Alamitos, CA, pp 285-292, 1991.

Darrell, T. et al., “Integrated Person Tracking Using Stereo, Color, and Pattern Detection” Conf. Computer Vision and Pattern Recognition, IEEE CS Press, Los Alamitos, Calif., pp. 601-608, 1998.

Di Gesu, V., Lenzitti, B., Lo Bosco, G. and Tegolo, D., “A Distributed Architecture for Autonomous Navigation of Robots”, Proceedings Fifth IEEE International Workshop on Computer Architectures for Machine Perception, pp.190 - 194, 2000.

Fernández, J. L. et al, “Improving collision avoidance for mobile robots in partially known environments: the beam curvature method”. Robotics and Autonomous Systems, Vol. 46, pp. 205-219, 2004.

Foley, J., Dam, A. V., Feiner, S., and Hughes, J., “Computer Graphics, principles and practice”. Addison Wesley edition 2nd, 1990.

Fox, D. et al, “The Dynamic Window Approach to Collision Avoidance”. IEEE Robotics and Automation Magazine 4 (1), pp. 23-33, 1997.

Fox, D., Burgard, W., and Thrun, S., “Markov localization for mobile robots in dynamic environments”, Journal of Artificial Intelligence BIBLIOGRAPHY 141 Research 11, pp 391-427, 1999.

Gabriely, Y. and Rimon, E., “Spanning-tree based coverage of continuous areas by a mobile robot”. Submitted to Annals of Mathematics and Artificial Intelligence.

Gallardo, D. and Colomina, O., “A Genetic Algorithm for Robust Motion Planing”, Eleventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Castellon, Spain, June, pp. 115-121, 1998.

Goto, Y., Stentz, A., “Mobile Robot Navigation: The CMU System,” IEEE Expert, Vol. 2, No. 4, Winter, 1987.

Harter, A. et al., “The Anatomy of a Context-Aware Application,” Proc. 5th Ann. Int’l Conf. Mobile Computing and Networking (Mobicom 99), ACM Press, New York, pp. 59-68, 1999.

Hightower, J. and Borriello, G., "Location Sensing Techniques", UW CSE Technical Report, 2001; available at:
www.cs.washington.edu/research/portolano/papers/UW-CSE-01-07-01.pdf

Hightower, J. and Borriello, G., "Location Systems for Ubiquitous Computing" Computer, vol. 34, no. 8, pp. 57-66, 2001.

Hightower, J., Want, R. and Borriello, G., “SpotON: An Indoor 3d Location Sensing Technology Based on RF Signal Strength”, UW CSE 2000-02-02, Univ. Washington, Seattle, 2000.

Hocaoglu, C. and Sanderson, A.C., "Planning Multiple Paths with Evolutionary Speciation", IEEE Trans. Evolutionary Computation, vol. 5, no. 3, pp. 169-191, 2001.

Hwang, Y.K., Ahuja, N., “Gross Motion Planning – A Survey”, ACM Computing Surveys, volume 24, issue 3, pp. 219-291, 1992.

Iida, S. and Yuta, S., “Vehicle Command System and Trajectory Control for Autonomous Mobile Robots”, IROS ‘91, Osaka, Japan. IEEE Cat. No. 91TH0375-6, 1991.

Jarvis, R. A., “Collision-Free Trajectory Planning Using the Distance Transforms,” Mechanical Engineering Trans. of the Institution of Engineers, Australia, Vol. ME10, No. 3, 1985.

Jenke, P., Huhle, B. and Straßer, W. “Self-Localization in Scanned 3DTV Sets”, in: 3DTV CON - The True Vision, 2007.

Khatib., O., “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots”. 1985 IEEE International Conference on Robotics and Automation, March 25-28, St. Louis, pp. 500-505, 1985.

Kubota, N., Morioka, T., Kojima, F. and Fukuda, T., “Perception-Based Genetic Algorithm for a Mobile Robot with Fuzzy Controllers”, Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), pp. 397-404, 1999.

Kurabayashi, D., Ota, J., Arai, T., and Yoshida, E., “Cooperative sweeping by mobile robots”. In IEEE International Conference on Robotics and Automation, pages 1744.1749, 1996.

Krumm, J. et al., “Multi-Camera Multi-Person Tracking for Easy Living” Proc. 3rd IEEE Int’l Workshop Visual Surveillance, IEEE Press, Piscataway, N.J., pp. 3-10, 2000.

Latombe, J.C., Robot motion planning, Kulwer Academic publishers, Boston, MA, 1991.

Lozano-Perez, T. and Wesley, M., “An algorithm for planning collision-Free paths among polyhedral obstacles,” Communications of the ACM, vol.22, pp.560-570, 1979.

Lu, F. and Milios, E., “Robot pose estimation in unknown environments by matching 2d range scans”. Intelligent and Robotic Systems, 18:249–275, 1997.

Lumelsky, V. J., Stepanov, A. A., “Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment”, IEEE Transactions on Automatic Control, Vol. AC- 31, No. 11, 1986.

McCleary, G. F. and Westbook, N., “Recreational and Recreational Mapping”, Sturbridge Village, 1974.

McFarland, D. and Bösser, T., “Intelligent Behaviour in Animals and Robots”, The MIT Press, 1993.

Minguez, J., Osuna, J., and Montano, L., “A divide and conquer strategy to achieve reactive collision avoidance in troublesome scenarios”. In International Conference on Robotics and Automation, Minessota, USA, 2004.

Minguez, J., Montesano, L., and Montano, L., “An Architecture for Sensor-Based Navigation in Realistic Dynamic and Troublesome Scenarios”. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan, 2004.

Minguez, J. and Montano, L., “Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios”. IEEE Transactions on Robotics and Automation, Vol. 20, No. 1, pp. 45-59, 2004.

Montesano, L., Minguez, J., Montano, L., “Lessons Learned in Integration for Sensor-Based Robot Navigation Systems”. International Journal of Advanced Robotic Systems, Volume 3, no. 1, Pages 85-91, 2006.

Moravec, H. P. and Elfes, A., “High Resolution Maps from Wide Angle Sonar”. IEEE Conference on Robotics and Automation, Washington D.C., pp. 116-121, 1985.

Nilsson, N. J., “Principles of Artificial Intelligence”, Tioga Publishing Company, 1980.

Obitko, M., “Genetic Algorithms”, Internet publication, 1998, http://cs.felk.cvut.cz/~xobitko/ga/main.html

Orr, R.J. and Abowd, G.D., “The Smart Floor: A Mechanism for Natural User Identification and Tracking,” Proc. 2000 Conf. Human Factors in Computing Systems (CHI 2000), ACM Press, New York, 2000.

Pratihar, D.K., Deb, K. and Ghosh, A. “Fuzzy-Genetic Algorithms and Mobile Robot Navigation among Static Obstacles”, Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), pp. 327-334, 1999.

Preparata, F.P. and Shamos, M.I., “Computational Geometry: An Introduction.” Springer-Verlag. p198-257, 1985.

Priyantha, N.B., Chakraborty, A. and Balakrishnan, H., “The Cricket Location-Support System,” Proc. 6th Ann. Int’l Conf. Mobile Computing and Networking (Mobicom 00), ACM Press, New York, pp. 32-43, 2000.

Raab, F. et al., “Magnetic Position and Orientation Tracking System,” IEEE Trans. Aerospace and Electronic Systems, pp. 709-717, 1979.

Ringbeck, T., Möller, T. and Hagebeuker, B. “Multidimensional measurement by using 3-D PMD sensors”, Adv. Radio Sci., 5, 135–146, 2007.

Schone, H., “Spatial Orientation: The Spatial Control of Behaviour in Animals and Man”, Princeton, NI: Princeton University Press, 1984.

Simmons, R.G., “The curvature velocity method for local obstacle avoidance”. Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 4, pp. 2275-2282, 1996.

Sugihara, K. and Smith, J., “Genetic Algorithms for Adaptive Motion Planning of an autonomous Mobile Robot”, Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA, pp. 138-146, 1997.

Technical Description of DC Magnetic Trackers, Ascension Technology Corp., Burlington, Vt., 2001.

Thrun, S., Bücken, A., Burgard, W., Fox, D., Fröhlinghaus, T., Hennig, D., Hofmann, T., Krell, M., and Schimdt, T.. ”Map learning and high-speed navigation in RHINO”. In D. Kortenkamp, R. Bonasso, and R. Murphy, editors, AI-based Mobile Robots: Case studies of successful robot systems. MIT Press, Cambridge, MA, 1998.

Thrun, S., Fox, D., Burgard, W. and Dellaert, F., “Robust monte carlo localization for mobile robots” Artificial Intelligence 128,1- 2, pp 99-141, 2001.

Ulrich, I. and Borenstein, J., “VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots”. Proceedings of the 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium, May 16-21, pp. 1572-1577, 1998.

Ulrich, I. and Borenstein, J., “VFH*: Local Obstacle Avoidance with Look-Ahead Verification”. 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, April 24-28, pp. 2505-2511, 2000.

Uribe, J. P. and Urzelai, J., “Fuzzy Controller for Obstacle Avoidance with a Non-Holonomous Mobile Robot”. Mathware & Soft Computing, Vol. 5, pp. 279-289, 1998.

Vadakkepat, P. and Chen, T.K., “Evolutionary Artificial Potential Fields and Their Application in Real Time Robot Path Planning”, Proceeding of the 2000 Congress on Evolutionary Computation, San Diego, CA, pp. 256-264, 2000.

Want, R. et al., “The Active Badge Location System” ACM Trans. Information Systems, pp. 91-102, 1992.

Xiao, J. and Zhang, L., “Adaptive Evolutionary Planner/Navigator for Mobile Robots”, IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 18-28, 1997.

Ye, C. and Borenstein, J., “Obstacle Avoidance for the Segway Robotic Mobility Platform”. American Nuclear Society 10th International Conference on Robotics and Remote Systems for Hazardous Environments, Gainsville, Fl, March 28-31, 2004.

Yuta, S., Suzuki, S. and Iida, S., “Implementation of a Small Size Experimental Self-Contained Autonomous Robot”, Proceedings of the 2nd Int. Symposium on Experimental Robotics, 1991.

Zelinsky, A., Jarvis, R. A., Byrne, J. C. and Yuta, S., “Planning paths of complete coverage of an unstructured environment by a mobile robot”. International Journal of Robotics Research, 13(4):315. 1994.