A Sensor Based Navigation Algorithm for a Mobile Robot using the *DVFF* Approach

A. OUALID DJEKOUNE[1], KARIM ACHOUR[1] and REDOUANE TOUMI[2]
[1]Robotics and Industrial Automation.
Advanced Technologies Development Centre.
Lotissement 20 Août 1956 Baba Hassen, BP17 Baba Hassen, Algiers, Algeria.
[2]Electronics Department, Engineering Faculty
U.S.T.H.B., BP. 32 Bab Ezzouar, 16111, Alger, Algeria.
odjekoune@cdta.dz.

Abstract: *Often autonomous mobile robots operate in environment for which prior maps are incomplete or inaccurate. They require the safe execution for a collision free motion to a goal position. This paper addresses a complete navigation method for a mobile robot that moves in unknown environment. Thus, a novel method called DVFF combining the Virtual Force Field (VFF) obstacle avoidance approach and global path planning based on D* algorithm is proposed. While D* generates global path information towards a goal position, the VFF local controller generates the admissible trajectories that ensure safe robot motion. Results and analysis from a battery of experiments with this new method implemented on a ATRV2 mobile robot are shown.*

Keywords: *Autonomous mobile robot, global navigation, VFF algorithm, obstacle avoidance, D* algorithm.*

1. Introduction

Mobile robot navigation systems require both sufficiently reliable estimation of the current robot location and precise map of the navigation area. These systems are separated into two levels of control: global path planning and local motion control. Path planning considers a model or a map of the environment to determine the geometric path points for the mobile robots to track from a start position to the goal. Whereas local motion usually use sensory information to determine a motion that will avoid collision with unknown obstacles or obstacles whose position in the environment had changed.
A variety of global path planning methods, such as road map, cell decomposition and potential field methods have been proposed. Their advantages lie in the fact that a complete trajectory from starting point to the goal point can be computed off line. However, they are not appropriate when the world map is inaccurate or unknown. The *A** algorithm is more used in these methods. It is a global search algorithm which gives a complete and optimal global path in static environments. It was improved in (Stenz, A., 1994) for efficient on line searching of dynamic environments. This algorithm named *D* search* is recognized as an effective global path searching method which returns sequences of path points in known or partially known environment.
The local navigation systems are capable of producing a new path in response to the environmental changes. These systems can be divided into directional and velocity space based approaches (Seder, M.; Macek, K.; Petrovic, I., 2005). The directional approaches such as Potential field method (Khatib, O. , 1986), Virtual Force Field (Borenstein, J. & Koren, Y., 1990) which extends to Vector Field Histogram (Borenstein, J. & Koren, Y., 1991) and Nearness Diagram algorithm (Minguez, J. & Montano, L, 2000), generate a direction for the robot to head in. Velocity space approaches such as Curvature Velocity method (Simmons, R., 1996), Lane Curvature method (Ko, N.Y. & Simmons, R., 1998), and Dynamic Window method (Fox, D.; Burgard, W. & Thrun, S., 1997), perform a search of the commands controlling the robot such as translational and rotational velocities directly from the velocities space.
A complete mobile robot navigation system should integrate the local and global navigation systems: the global system pre-plan a global path and incrementally search new paths when discrepancy with the map occurs; the local system uses onboard sensors to detect and avoid unpredictable obstacles. The mobile robot executes an algorithm which permits it to follow the optimal global path by tracking its geometric points from a start position to the goal. If an obstacle obstructs this path, the robot executes another algorithm (collision avoidance algorithm) allowing it to move around the perimeter until the nearest point of the obstacle to the geometric path point is found, or pre-plan another optimal global path to reach the goal.
In this paper, a complete mobile robot navigation method called *DVFF* is introduced. It combines a global path planning based on *D** algorithm, with a real time obstacle avoidance algorithm based on *VFF* approach proposed in (Oualid Djekoune, Karim Achour & Redouane Toumi , 2005). The originality of this method lies in the use of the

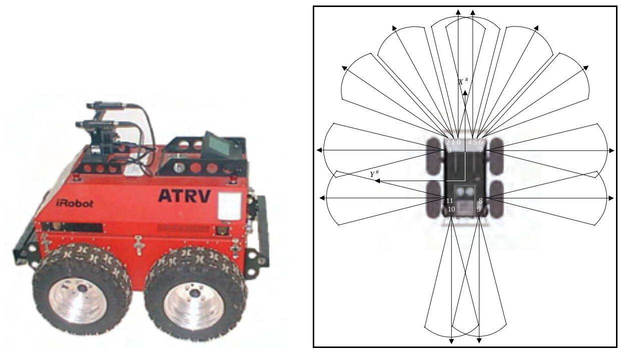
*D** algorithm only to generate the global path information. The global path information are the backpointer directions. The optimal path from any position in the environment can be determined simply by the following global path information to reach the goal. The mobile robot can reach the goal without carrying out a path tracking algorithm. That permits to the robot to save the time wasted in the path tracking operation.
The map is built with two dimensional Cartesian histogram grid based on a modified Histogramic In Motion Mapping (*HIMM*) algorithm, and updated continuously with range data sampled by onboard ultrasonic sensors. This algorithm does not use C-space to enlarge the cells in the map prior to path search algorithm to account for robot dimensions, and allows adding and retrieving data on the fly and enables an easy integration of multiple sensors.

2. Prior Work

Nearly all researches in a complete navigation system combine a global path planning module with a local obstacle avoidance module to perform navigation. While the global path planner determines a suitable path based on a map of the environment, the obstacle avoidance algorithm determines a suitable direction of motion based on recent sensor data. Obstacle avoidance is performed locally in order to ensure that real-time constraints are satisfied.
For example the Vector Field Histogram* algorithm (*VFH**) developed in (Borenstein Johann & Ulrich Iwan, 2000), combines the *VFH+* (Borenstein Johann & Ulrich Iwan, 1998) as local obstacle avoidance algorithm with the *A** search algorithm as global path planner. It is considered as a local obstacle avoidance algorithm that uses look-ahead verification to consider more than the robot's immediate surroundings. While *VFH** has the same obstacle avoidance performance as *VFH+* for regular obstacles, *VFH** is capable of dealing with problematic situations that would require the robot to substantially slow down or even stop (Borenstein Johann & Ulrich Iwan, 2000). The *VFH+* is a real-time local obstacle avoidance algorithm that looks for gaps in locally constructed polar histograms.
In the same way, several well known local obstacle avoidance such as Dynamic Window (*DW*) and Nearness Diagram (*ND*) are combined with a global path planner to perform navigation and to become respectively Global Dynamic Window (*GDW*) and Global Nearness Diagram (*GND*) complete navigation systems. They are combined with the global, local minima-free navigation function NF1 (Oliver Brock & Oussama Khatib, 1999) using a wave propagation technique starting at the goal. As used in (Macek, K. and Petrovic, I. & Ivanijko, E., 2003), a complete navigation system was developed which integrates the *A** algorithm and the *DW* algorithm and then adapted for the use of focused *D** algorithm (Seder, M.; Macek, K.; Petrovic, I., 2005) instead of *A** algorithm.
The use of global reasoning in the above approaches is essentially to overcome the shortcomings of the used local obstacle avoidance algorithm.
In (Oualid Djekoune, Karim Achour & Redouane Toumi, 2005) a real time obstacle avoidance based only on the *VFF* approach have been presented and tested on a real *ATRV2* mobile robot. However, based on several experiments, some shortcomings that are inherent to the concept of potential fields are discovered, the mobile robot was gotten trapped in local minima when it entered a dead end. If the mobile robot has a global knowledge of the goal position, it would be easy for it to go out from the local minima. This global knowledge can be obtained using a global path planner.
The *D** search algorithm is currently most widely used in partially known or changing environment. It has been shown to be one to two orders of magnitude more efficient than planning from scratch with *A** (Dave Ferguson & Anthony Stentz, 2005). It produces an optimal path from the start position to the goal by minimizing a predefined cost function. It has the capability of rapid replanning, and has been used in real time planning in partially known environment with challenging terrains.
With the *D** algorithm, we can obtain the global knowledge of the goal position from any position in the environment. The global knowledge is called in our case the global path information are the backpointer direction. The backpointers are determined from cost calculation to all positions in the robot space navigation. The optimal path from any position in the environment can be determined by the following global path information to reach the goal.
In addition, *D** algorithm can be easily combined with the real time obstacle avoidance algorithm developed in (Oualid Djekoune, Karim Achour & Redouane Toumi, 2005) because first, they both generate a direction, and second it does not use necessary a C-space to enlarge the cells in the map grid according to the robot dimensions when we use local obstacle avoidance algorithm. The mobile robot can either use the resultant of these two directions or one of them to avoid detected obstacles or to reach the goal.

3. Scene Map Building Algorithm

In order to create a scene map from ultrasonic range measurements, the environment must be scanned at first. The mobile robot *ATRV2* (Fig.1.a) is equipped with 12 ultrasonic sensors that are mounted on a horizontal ring around the robot (six on the front, two on the back and two on both sides). Their measuring range spans distances from approximately 5 centimeters to 4 meters. The main lobe of the sensitivity function is contained within a solid angle Ω of 30° (Fig.1.b).
The sonar devices are Polaroid transducers, widely used in robotics to avoid obstacles, range sensing and map building. These sensors are low cost and their signals



(a) (b)

Fig. 1. (a) *ATRV2* mobile robot, (b) the sensing coverage around the *ATRV2.*

have a reduced bandwidth, enabling the use of complex processing algorithms to obtain real time information about the surrounding objects.
With the robot's ultrasonic sensors, an approximate full 360° panorama can be acquired rapidly. However, all sensors are fired at once; unfortunately, this causes significant crosstalk (caused by low angular resolution and errors due to multiple reflections or specular reflections away from the sensor).

3.1. Histogram grid for obstacles representation

The map building algorithm presented here is an *HIMM* presented in (Borenstein, J. & Koren, Y., 1991) modified to improve our approach. It uses a two dimensional Cartesian histogram grid for obstacle representation. The *HIMM* algorithm is derived from the certainty grid concept described in (Borenstein, J. & Koren, Y., 1990), and differs from this last one in the way it is built and updated. This algorithm is especially suited to the unified representation of data from different sensors such as ultrasonic, vision and proximitry sensors (Moravec, H. P., 1988).
Like the certainty grid, each cell in the histogram grid holds a *Certainty Value "CV"* that represents the algorithm confidence in the existence of an obstacle at that location. In (Borenstein, J. & Koren, Y., 1991) the *CVs* are increased or decreased by sensor data until predefined maximum or minimum values are reached, only for cells which lie on the acoustic axis of the ultrasonic sensor. This algorithm ignores the angular error and supposes that the detected object is closer to the acoustic axis of the sensor than the periphery of the view conical field However, while considering that without complementary information on ultrasonic measure, all points of the cone's periphery are equivalent for a possible position of the obstacle; It appears to us that it is necessary to assign the same value to each cell representing the same state (free or occupied).
Our method uses this idea by decreasing by 1 the *CV* of cells corresponding to the free space until a predefined minimum value is reached, and increasing by 1 the *CV* of cells of the cone's periphery representing the obstacle until a predefined maximum value is reached. The *CVs* of the remaining cells are not modified.
Initially, the histogram grid world model is set to the 0 value. The *CV* cells values of the known obstacles are set by a predefined maximum value. These values change if the obstacle can move and preserve their values otherwise (walls, etc.). When the robot moves, each detected obstacle increases the value of the corresponding cell. Progressively, while the robot progress; the obstacles are identified by cells whose values increase. That allows the robot to avoid them.
This method has the advantage of avoiding the measurements treatment using probabilities laws which are heavy to manage.

3.2. Real time scene map building

A local map is associated with the robot. Its center is the gravity center of the mobile robot. Its size is adjusted on the largest detected ultrasonic measure. At any time instance, we have a previous map associated with the robot at the last position, containing all the processed sensing information so far within the local map. Along with this previous map, we have the most recent data set that is assumed to be collected at the current robot position. We process the current data set within the current local map using an update rule described in the previous paragraph. While the robot evolves, all local maps are integrated into a one global map.

4. The D^* Algorithm For Global Path Information Processing

The path planning problem of a mobile robot is to find a safe and an efficient path for the mobile robot, given a start position, a goal position and a map of the workspace. The robot can go from the start position to the goal without colliding with any obstacle along the path.
Generally, a robot does not have complete map information. As a result, any path generated using its initial map may turn out to be invalid or suboptimal as it receives updated map information through its onboard sensors. It is thus important that the robot is able to update its map and replan optimal paths when new information arrives (Dave Ferguson & Anthony Stentz, 2005).
A number of algorithms exist for performing this replanning (Stenz, A., 1994) (Stentz, A., 1995) (Barbehenn, M. & Hutchinson, S., 1995) (Ramalingam, G. & Reps, T., 1996) (Ersson, T. & Hu, X., 2001) (Huiming, Y.; Chia-Jung, C.; Tong, S. & Qiang, B., 2001) (Podsedkowski, L.; Nowakowski, J.; Idzikowski, M.& Vizvary, I., 2001) (Koenig, S. & Likhachev, M., 2002). *A** and *D** are currently the most widely used of these algorithms, due to their efficient use of heuristics and incremental updates. *D** has been shown to be one to two orders of magnitude more efficient than planning from scratch with *A** which has been incorporated into real robotic systems. These algorithms guarantee an optimal paths over grid-based representations of a robot's environment (Dave Ferguson & Anthony Stentz, 2005). The *D** search

algorithm is a dynamic version of *A**. It produces an optimal path from the start position to the goal by minimizing a predefined cost function. It has the capability of rapid replanning, and has been used in real time planning in partially known environment with challenging terrains.
In our approach, we have used *D** algorithm because, it allows replanning to occur incrementally and optimally in real time (Stenz, A., 1994). It also gives the global path information from any position in the environment towards the goal. Applied on a map represented by two dimensional Cartesian histogram grid, each cell of the map includes an estimate of the path cost to the goal, and a back pointer to one of its neighbours indicating the geometric direction to the goal (north, south, east, west, north-west, south-west, north-east and south-east) called in our case the global path information. The figure 2 shows how to calculate the global path information according to the back pointer position.
This information is very important for a mobile robot moving in the environment. From any position (or the map cell), the mobile robot can have the information indicating to it the global steering direction toward the goal. The figure 3 shows the global path information result of a simulated obstacles course with a given start and goal positions.
Following this direction, the mobile robot can reach the goal without it carrying out a path tracking algorithm. That permits the robot saving time wasted in the path tracking operation.
In the presented work, the mobile robot uses the back pointer of the cell under the point *CP_1* located on the longitudinal axis of the robot. Its optimal location differs for different mobile robots (Johann Borenstein & Ulrich Raschke, 1991). The robot determines the global path information θ_G (*CP_1* back pointer direction) and applies a fictitious vector F_{D*} in that direction (Fig.4). This direction will be used to calculate the motion command that generates a collision free motion while simultaneously driving the robot towards the goal.

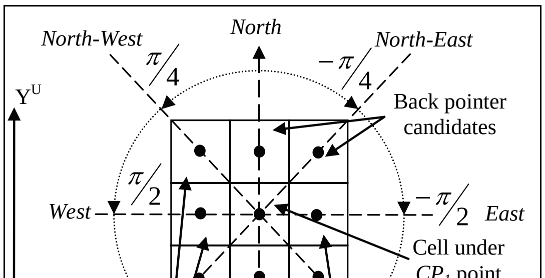


Fig. 2. Computing the global path information under *CP_1* point.

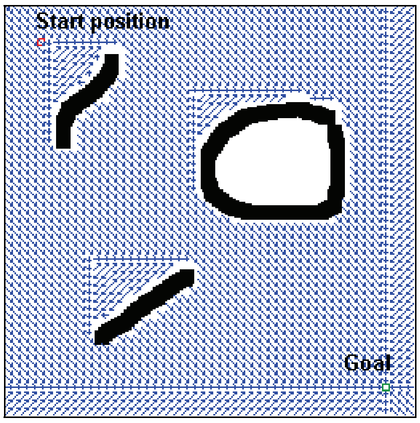


Fig. 3. Global path information from any position in the environment.

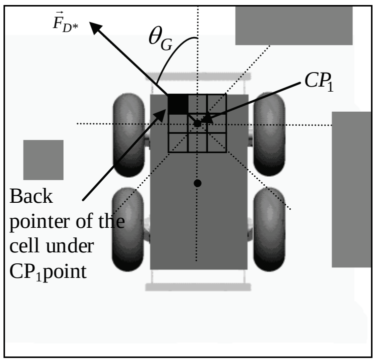


Fig. 4. Computing the global steering direction.

In addition to the certainty values used in the *HIMM* algorithm (see section 3), some attributes were added to each cell of the grid when we use the *D** algorithm such as *back pointer (b)*, *arc cost (c)*, *tag (t)*, *path cost (h)*, *key (k)*. For the detected obstacles, we have used the *walkable* or traversable attribute which will be used by the developed algorithm to ensure its fast convergence
The obstacles sizes are not enlarged by a safety margin. The cells of the grid surrounding (according to the robot dimensions) these obstacles will not be used by the D^* algorithm.
The cost of traversing from cell *Y* to *X* is a positive number given by the *arc cost* function $c(X,Y)$ by the following equation:

$$c(X,Y)=\begin{cases}10 & \textit{if Y is situated horizontally} \\ & \textit{or vertically to X} \\ 14 & \textit{if Y is situated on} \\ & \textit{the diagonal to X} \\ \textit{X and Y are neigbors} & \end{cases} \quad (1)$$

4.1. Description of the algorithm

The used D^* is inspired from the work presented in (Stenz, A., 1994) and modified to improve our approach. It consists primarily of two functions: *Init_D^** and *Case_D^**. *Init_D^** is used only at the beginning of the algorithm to compute the initial optimal path from the start position to the goal (*G*), and Case_D^* is used when a new obstacle is detected by the mobile robot (arc cost function of a state is changed) to compute the new optimal path from the robot position to the goal.

Initially, all cell tag values are set to *NEW*, *h(G)* is set to zero, and *G* is placed on the *OpenList*. The first function, *Init_D** is repeatedly called until the start position is removed from the *OpenList* or a value of -1 is returned, at which point either a sequence of path points has been computed or does not exist. The mobile robot then proceeds to follow the backpointers in the computed sequence until either it reaches the goal point or discovers an error in the arc cost function (e.g., due to a detected obstacle). The second function, *Case_D**, is immediately called to correct and to compute a new sequence, and the mobile robot continues to follow the backpointers in the new sequence toward the goal.

The algorithms for *Init_D** and *Case_D** are presented below. We have presented these algorithms in the same framework as that used by (Stenz, A., 1994) to highlight both its differences and similarities.

The embedded routines are *MinState* , which returns the state on the *OpenList* with minimum *k* value (*Null* if the list is empty); *GetKmin*, which returns k_{min} for the *OpenList* (-1 if the list is empty); *Delete(X)*, which deletes the state *X* from the *OpenList* and sets *t(X)=Closed* ; ind *Insert(X,h_{new})*, which computes $k(X)=h_{new}$ if $t(X)=New$, $k(X)=min(k(X),h_{new})$ if $t(X)=Open$, and $k(X)=min(h(X),h_{new})$ if $t(X)=Closed$, sets $h(X)=h_{new}$ and $t(X)=Open$, and places or repositions state *X* on the *OpenList* sorted by *k* value (for more details consult (Stenz, A., 1994)).

Function : Init_D* .

```
L1  Do{
L2    X = MinState().
L3    if (X == Null)  break.
L4    k_old = GetKmin().
L5    Delete(X).
L6    wc = WorkableCase(X).
L7    if(wc){
L8      if(k_old ≺ h(X))
L9        ProcessI(X, k_old).
L10     else if(k_old == h(X))
L11           ProcessII(X, k_old).
L12     else ProcessIII(X, k_old).
L13       }
L14   k_p = GetKmin().
L15   }while(k_p != -1).
```

Function : Case_D* (X_{Rob}).

```
L1  The tag value of all states is set to New.
L2  The backpointer of all states is set to -1.
L3  The goal G in the OpenList.
L4  Do{
L5    X = MinState().
L6    if ((X == Null) or (X == X_Rob))  break.
L7    k_old = GetKmin().
L8    Delete(X).
L9    wc = WorkableCase(X).
L10   if(wc){
L11     if(k_old ≺ h(X))
L12       ProcessI(X, k_old).
L13     else if(k_old == h(X))
L14         ProcessII(X, k_old).
L15     else ProcessIII(X, k_old).
L16       }
L17   k_p = GetKmin().
L18   }while(k_p != -1).
```

Function : ProcessI(X, k_{old}).

```
L1  for each neighbor Y of X :
L2    if ((h(Y) ≤ k_old) and (h(X) ≻ (h(X) + c(Y, X)))){
L3      b(X) = Y;
L4      h(X) = h(Y) + c(Y, X).
L5                                  }
```

Function : ProcessII(X, k_{old}).

```
L1  for each neighbor Y of X :
L2    if ((t(Y) = New) or
L3      (b(Y) == X) and (h(Y) != (h(X) + c(X,Y)) or
L4      (b(Y) != X) and (h(Y) ≻ (h(X) + c(X,Y)))){
L5        b(Y) = X.
L6        Insert(Y, h(X) + c(X,Y)).
L7                                  }
```

bool WorkableCase(X).

```
L1  wc = true.
L2    for each neighbor Y of X :
L3      if(Y is not walkable or
L4          Y's CV ≻ 0)   wc = false.
L5  return(wc).
```

Function : ProcessIII(X, k_{old}).

```
L1  for each neighbor Y of X :
L2    if ((t(Y) = New) or
L3      (b(Y) == X) and (h(Y) != (h(X) + c(X,Y)))){
L4        b(Y) = X.
L6        Insert(Y, h(X) + c(X,Y)).
L7                                  }
L8    else if((b(Y) != X) and (h(Y) ≻ (h(X) + c(X,Y)))
L9        Insert(X, h(X)).
L10   else if((b(Y) != X) and (h(X) ≻ (h(Y) + c(Y, X)))
L11         and (t(Y) = Closed) and (h(Y) ≻ k_old)){
L12         Insert(Y, h(Y)).
L13                                 }
```

5. *VFF* Approach for Real Time Obstacle Avoidance

Real time obstacle avoidance presents the problem of navigating around known or unknown objects in a dynamic environment (Chris Gourley & Mohan M. Trivedi, 1994). The obstacle avoidance algorithm implemented in this work is the extension to the one developed in (Oualid Djekoune, Karim Achour & Redouane Toumi, 2005). It is similar to the concept described in (Johann Borenstein & Ulrich Raschke, 1991), but varies in that it only uses the *VFF* method to calculate both the frontal and the lateral corrective measures without using the *VFH* (Vector Field Histogram) approach. These measures are used to calculate the local steering direction to protect the robot from collisions with the detected obstacles.

5.1. Computing the frontal repulsive force

The *VFF* method is applied at the point *CP₁* to determine the frontal repulsive force applied to the robot, pushing the robot away from the detected obstacles. It uses the measures obtained from the six ultrasonic sensors situated at the front of the mobile robot (Fig.5).

The resultant repulsive force $\vec{F}_F$ is the vectorial sum of the individual forces from all frontal occupied cells, it is given by:

$$\vec{F}_F = \sum_i \frac{F_{cr}\, C_i}{d_i^2}\left[\frac{x_i - x_c}{d_i}\vec{i} + \frac{y_i - y_c}{d_i}\vec{j}\right] \tag{2}$$

Where F_{cr} is the force constant (repelling), d_i the distance between occupied cell *i* and the cell under the point *CP₁* of the robot, C_i the certainty value of the occupied cell *i*, (x_c, y_c) the cell coordinates under the point *CP₁* of the robot, and (x_i, y_i) the coordinates of the occupied cell *i*.

5.2. Computing the lateral repulsive force

To protect the mobile robot from any lateral collisions, the *VFF* method is applied to each one of the sensors on either side of the mobile robot using the following equation:

$$\vec{F}_s = \frac{F_{cr}\, C_i}{d_i^2}\left[\frac{x_i - x_s}{d_i}\vec{i} + \frac{y_i - y_s}{d_i}\vec{j}\right] \tag{3}$$

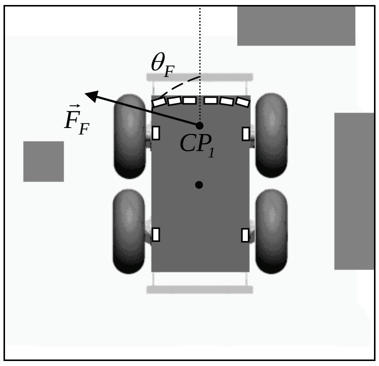


Fig. 5. Computing the frontal repulsive force $\vec{F}_F$.

Where F_{cr} is the force constant (repelling), d_i the distance between the occupied cell *i* and the sensor *s*, C_i the certainty value of the occupied cell *i*, (x_s, y_s) the lateral sensor coordinates, and (x_i, y_i) the coordinates of the occupied cell *i*.

Applying the principle of *free-body diagrams* as in (Johann Borenstein & Ulrich Raschke, 1991), all these forces are replaced by a single lateral force $\vec{F}_L$ and a moment *M*, acting on the robot at the center point *CP* (Fig.6). Next, they are decomposed into a force couple $(\vec{F}_{1m}, \vec{F}_{2m})$ and two forces $(\vec{F}_{1f}, \vec{F}_{2f})$ (Fig.7). $\vec{F}_{1m}$ acts on *CP₁* and $\vec{F}_{2m}$ acts on a symmetrically located point *CP₂* in the rear part of the mobile robot. Note that the force couple $(\vec{F}_{1m}, \vec{F}_{2m})$ is statically equivalent to the moment *M* if it is computed as:

$$F_{1m} = F_{2m} = \frac{M}{d} \tag{4}$$

where *d* is the distance between *CP* and *CP₁* (or *CP₂*).

The two forces $(\vec{F}_{1f}, \vec{F}_{2f})$ are acting on *CP₁* and *CP₂*, respectively and verifying the following equation:

$$F_{1f} = F_{2f} = \frac{F_L}{2} \tag{5}$$

The forces $\vec{F}_{2m}$ and $\vec{F}_{2f}$ are not considered in our case because the used mobile robot has only two degrees-of-freedom (Johann Borenstein & Ulrich Raschke, 1991).

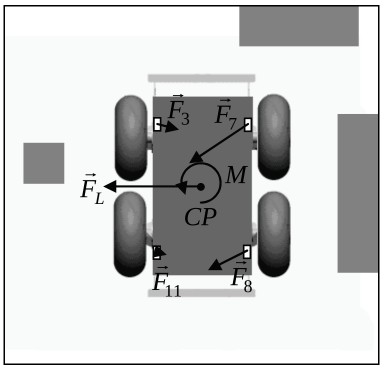


Fig. 6. Computing the lateral repulsive forces.

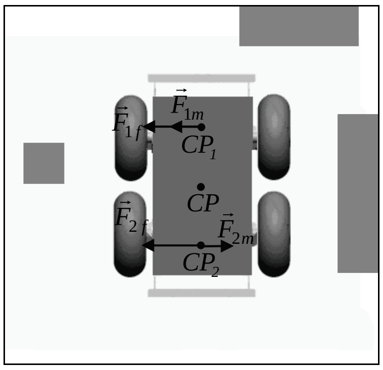


Fig. 7. The lateral computed forces decomposition.

The force $\vec{F}_{Rep}$ represents the corrected repulsive force on the robot, it pushes the robot away according to the direction θ_{Rep} from the frontal and lateral detected obstacles. It is the vectorial sum of all vectors acting on CP_1 point and is given by (Fig.8):

$$\vec{F}_{Rep} = \alpha \, \vec{F}_{1f} + \beta \, \vec{F}_{1m} + \gamma \, \vec{F}_F \tag{6}$$

Where the coefficients α , β and γ are set experimentally. The choice of these parameters is very significant, because they influence the displacements of the mobile robot, the rotational displacement around the detected obstacles and the translational displacement toward the goal.

6. The *DVFF* Navigation Algorithm

In most of the developed complete navigation systems, the mobile robot follows the optimal global path. If an obstacle obstructs this path, the robot moves around the perimeter until the nearest point of the obstacle to the goal is found (Lumelsky, V. J., Stepanov, A. A., 1986), or it uses a reactive collision avoidance with obstacles such as *DW* technique, then it plans a new optimal path to follow toward the goal (Seder, M.; Macek, K.; Petrovic, I., 2005).

Generally these systems combine local and global navigation systems. Sometimes, this combination is not simple and requires a large amount of computing power. For example, the *VFH** developed in (Borenstein Johann & Ulrich Iwan, 2000) is applied in static environment. It combines the *VFH+* (Borenstein Johann & Ulrich Iwan, 1998) as local obstacle avoidance algorithm with the *A** search algorithm as global path planner to analyze the consequences of heading towards each primary candidate direction. The primary candidate directions are calculated according to the generated global path before making a final choice for the new direction of motion. The inconvenient of this method is that their performances are at the expense of computational time (Borenstein Johann & Ulrich Iwan, 1998).

Another complete navigation system applied in a non static environment was developed in (Seder, M.; Macek, K.; Petrovic, I., 2005) which integrates focused *D** graph search algorithm for generation global geometric path and *DW* module for generation of possible robot trajectories called effectives path. The effective path is determined according to detection of a reference point on the global geometric path where the path direction starts changing significantly by observing path direction change points along the path. The length and orientation of the effective path directly determines optimal reference velocity vector in the next sampling instant that is a combined objective of obstacle clearance and path alignment. This navigation system has the same technique as the first one. They calculate the new direction of motion from some possible local trajectories according to the generated global path.

Generally, the *D** graph search algorithm is used only to generate a global path from the start position in the environment to the goal. It gives also an estimate of the path cost to the goal from any state of the graph and a back pointer to one of its neighbours indicating the geometric direction to the goal.

In our approach, we are interested in the backpointer information called in our case the global path information. This information indicates the geometric direction to the goal (north, south, east, west, north-west, south-west, north-east and south-east) from each cell or position in the robot environment. Following those directions, the mobile robot can reach the goal without carrying out a path tracking algorithm (Fig.2 and 3).

As detailed above (see section 4), some cells don't have a backpointer because they surround obstacles (according to the robot dimensions). In this case, the mobile robot will not have any global information which permits it reaching the goal.

In order to overcome this shortcoming, a local navigation approach is used. We have used a real time local obstacle avoidance algorithm based on *VFF* approach proposed in (Oualid Djekoune, Karim Achour & Redouane Toumi , 2005) and detailed above in the section five.

The combination of the algorithm proposed in (Oualid Djekoune, Karim Achour & Redouane Toumi , 2005) with the one detailed in the section four is easy because they generate both a direction. The first one generates a local direction that ensures safe robot motion, and the second one generates a global path information (or global direction) toward the goal which permits it to reach the goal. These directions can be used separately (global or local) or combined to give one direction. The selected direction will be used to calculate the motion command that generates a collision free motion while simultaneously driving the robot towards the goal

Unlike for complete navigation systems described above, this new navigation system has the advantage that its local obstacle avoidance algorithm generates only one direction which either can be considered only or combined with the global path information for the navigation.

In (Djekoune Oualid A., Karim Achour & Redouane Toumi, 2007) the authors show that when a mobile robot uses the resultant of these two directions, it will be able to navigate without collisions with the obstacles. The experimental tests with a real *ATRV2* mobile robot have proved to be successful except if the robot is in a local minimum. The authors use two coefficients for each direction. If they increase only the value of the coefficient used for the global direction, the mobile robot reaches the goal but avoids badly the obstacles. And if they increase only the value of the coefficient used for the local direction, the mobile robot avoids well the obstacles but the trajectory is not optimal. The choice of these coefficients isn't simple because they must be chosen experimentally (Djekoune Oualid A., Karim Achour & Redouane Toumi, 2007).

To overcome this shortcoming, the mobile robot will use separately these two directions. The mobile robot uses the global path information from the global path planning algorithm if it exists (the mobile robot is far from the obstacles). At any position in the environment, the robot uses the global information (the back pointer direction) of the cell under its point CP_1 to have the direction θ_G to the goal. This direction is used to calculate the motion command to move the robot towards the goal. As the robot moves, the obstacle avoidance module scans the local environment for unknown obstacles. This module does not affect the navigation if the global path information exists.

When the mobile robot is near either to a frontal or to a lateral detected obstacle, the cells surrounding these obstacles do not have the global path information. In this case, the obstacle avoidance algorithm calculates a local direction θ_{Rep} to generate a collision free motion to move the robot around the detected obstacles. Additionally, this module updates the global map of the environment for calculating the new global path information from its actual position towards the goal if the detected obstacles are new.

When the robot moves and at any time instance, the mobile robot follows the global path information to the goal if it exists, otherwise the local steering direction is used to protect the robot from collisions with the detected frontal or lateral obstacles. Each of these two directions is used to calculate the motion command that generates a collision free motion for the next sampling period *T*, while simultaneously driving the robot towards the goal.

The below algorithm illustrates in details how the mobile robot combines local and global information in a complete navigation way to permit a safe navigation from an initial position to a goal position.

```
Combinaison of global and local information algorithm :
L1  Compute the initial optimal path using the Init_D* function.
L2  The mobile robot starts to follow the global path information $\theta_G$.
L3  While the mobile robot moves and while the goal is not reached, it :
L4          Reads the odometry data.
L5              If the target is reached, the mobile robot stops.
L6          Reads the US data.
L7              If new obstacles are detected, the mobile robot uses the Case _ D* function.
L8          Reads the the global path information $\theta_G$ under its $CP_1$ point.
L9              If $\theta_G$ exists, the mobile robot uses this direction to move.
L10             Else, it calculates the local direction $\theta_{Rep}$ to move.
```

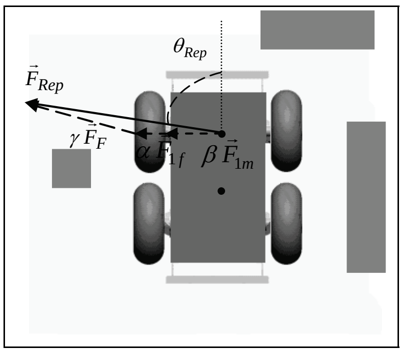


Fig. 8. Computing the corrected steering direction.

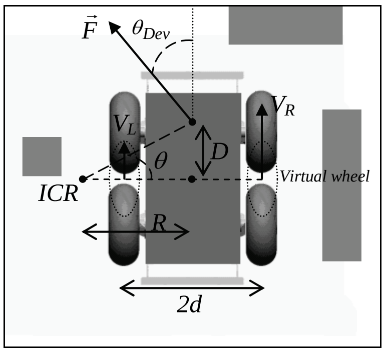


Fig. 9. The *ATRV2* kinematic model and its motion command.

7. Motion Control

The mobile robot *ATRV2* is a four wheeled differential drive skid steering configuration. The two wheels on the same side move in unison, with each pair on opposite sides capable of being driven independently. It can be seen like a robot with one virtual wheel on each side and castors (Fig.9).

The kinematic model of a two driving wheeled platform can be expressed by the following equation:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} w \tag{7}$$

where $(\dot{x}, \dot{y})$ is the Cartesian velocity, and (v, w) the linear and angular velocity.

The velocities *v* and *w* are expressed as follow:

$$\begin{cases} v = \dfrac{v_R + v_L}{2} \\ w = \dfrac{v_R - v_L}{2d} \end{cases} \tag{8}$$

where v_R and v_L are the linear velocities of the right and left wheels.

If both wheels are driving forward with the same speed, the robot moves forward. If they are driving in opposite directions, the robot will hovers around itself, but if they are driving forward with different speeds, the robot will follow a curve on the side moving with the lower speed. In this case, the mobile robot describes an arc of circle of radius *R* around the Instantaneous Center of Rotation (ICR) (Fig.9). *R* can be written as follow:

$$R = d\,\frac{v_R + v_L}{v_R - v_L} \tag{9}$$

When the desired steering direction θ_{Dev} (θ_{Rep} or θ_G) obtained from the navigation algorithm detailed above and the mobile robot orientation θ_{Rob} are known, the steering direction θ_c given to the robot is processed as follows:

$$\begin{aligned} &\Delta\theta = \theta_{Dev} - \theta_{Rob} \\ &if\left(-\pi \le \Delta\theta \le \pi\right)\ \theta_c = \Delta\theta. \\ &else\,if\ \left(\Delta\theta \prec -\pi\right)\ \theta_c = 2\pi + \Delta\theta. \\ &\quad else\,if\left(\Delta\theta \succ \pi\right)\ \theta_c = \Delta\theta - 2\pi. \end{aligned} \tag{10}$$

The desired respective linear and angular velocities v_d and w_d, can then be computed as follows according to the steering direction θ_c :

$$if\left(-\pi/180 \le \theta_c \le \pi/180\right)\begin{cases} move\ straight\ on. \\ v_d = v_c. \\ w_d = 0. \end{cases}$$

$$else\,if\left(\pi/180 \le \theta_c \prec \pi/2\right)\begin{cases} turn\ on\ the\ left. \\ R = \dfrac{D}{tg\ \theta_c}. \\ v_d = \dfrac{v_c}{2}\left(1+\dfrac{R-d}{R+d}\right). \\ w_d = \dfrac{v_c}{2d}\left(1-\dfrac{R-d}{R+d}\right). \end{cases}$$

$$else\,if\left(-\pi/2 \prec \theta_c \le -\pi/180\right)\begin{cases} turn\ on\ the\ right. \\ R = \dfrac{D}{tg\ \theta_c}. \\ v_d = \dfrac{v_c}{2}\left(1+\dfrac{R+d}{R-d}\right). \\ w_d = \dfrac{v_c}{2d}\left(\dfrac{R+d}{R-d}-1\right). \end{cases}$$

$$else\,if\left(\theta_c = \pi/2\right)\begin{cases} turn\ on\ the\ left. \\ v_d = 0. \\ w_d = \dfrac{v_c}{d}. \end{cases}$$

$$else\,if\left(\pi/2 \le \theta_c \le 175\pi/180\right)\begin{cases} turn\ on\ the\ left\ then \\ straight\ on \\ v_d = 0. \\ w_d = \dfrac{v_c}{d}. \end{cases}$$

$$else\,if\left(-175\pi/180 \le \theta_c \le -\pi/2\right)\begin{cases} turn\ on\ the\ right\ then \\ straight\ on \\ v_d = 0. \\ w_d = -\dfrac{v_c}{d} \end{cases}$$

$else\ Blocking\ \ Situation..$

where the constant v_c is set experimentally.

The value of the steering direction θ_c determines how the mobile robot moves. If this value is between $\frac{-\pi}{2}$ and $\frac{\pi}{2}$, the mobile robot describes an arc of a circle with radius *R*, around its *ICR* point. According to the *R* value and its sign, the mobile robot can either move forward or follow a curve. If this value borders π or $-\pi$, the mobile robot is on a blocking situation. In this case, the mobile robot calls the function *Case_D** to replan a new optimal path to go towards the goal. Otherwise, the robot turns in place then moves forward.

8. Experimental Results

The complete mobile robot navigation algorithm called *DVFF* presented in this paper has been implemented and tested on the *ATRV2* mobile robot manufactured by *iRobot* (Figure 1.a). They are written in the *C* language and using the robot software platform (*mobility* from *RWI*) under Linux.

This robot is mainly designed for outdoor applications and research, including four wheels, a stereo pair camera (not used in this framework), four odometers, one on each wheel, twelve ultrasonic sensors mounted on a horizontal ring around the robot, and a *Pentium III* on board computer under Linux serves as the robot's low level controller.

Five trials runs that illustrate different navigations of the *ATRV2* mobile robot through difficult obstacle courses are examined in this section.

To test the performance limits of the developed *DVFF* navigation method, in all those navigations, the environment and the obstacle locations were completely unknown and carried out in our laboratory. Both of the start position and the goal position were given only in advance to the robot. The translation velocity was limited to $v_c = 20\,cm/s$ and the rotational velocity was limited to $0{,}6\,rd/s$. The experiments are shown in the figures 11 to 14.

In each experiment, initially the mobile robot calls the function *Init_D** to compute the initial optimal path from the start position to the goal. The figure 10 shows the result of this function. The red dots show the cells in the histogram grid that the robot must have to reach the goal. The mobile robot uses then the global path information of the cell under its point CP_1 if it exists to have the direction θ_G to follow until it arrives to the goal. If θ_G does not exist, in this case the mobile robot is initially nearest to a unknown frontal or lateral obstacles, it computes then the direction θ_{Rep} of the corrected repulsive force which will allow it going away from these obstacles.

As the mobile robot drives, it receives information through its onboard ultrasonic sensors. If an obstacle obstructs the path of the robot, the robot calls the function *Case_D** to replan a new optimal path to permit it moving around this obstacle and going towards the goal. The new information received by the sensors is used both, to update the map of the environment and to process new global path information using the function *Case_D**.

The first experiment is shown in Fig.11. In this experiment, the start position and the goal position were taken aligned and any obstacles lie between those two positions. The function *Init_D** computes an initial optimal path like shown in the figure 10, then the robot starts to follow it until it reaches the goal. The reproduced histogram grid of this experiment is shown in fig.11.b. The empty cells are not shown, while filled cells are represented by small blue or red rectangles. The red rectangles indicate the cells crossed by the gravity center of the mobile robot. The bleu rectangles indicate the cells occupied by the obstacles.

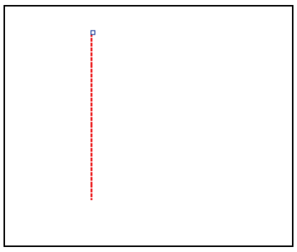


Fig. 10. The initial optimal path from the start position to the goal calculated before any displacement of the mobile robot.

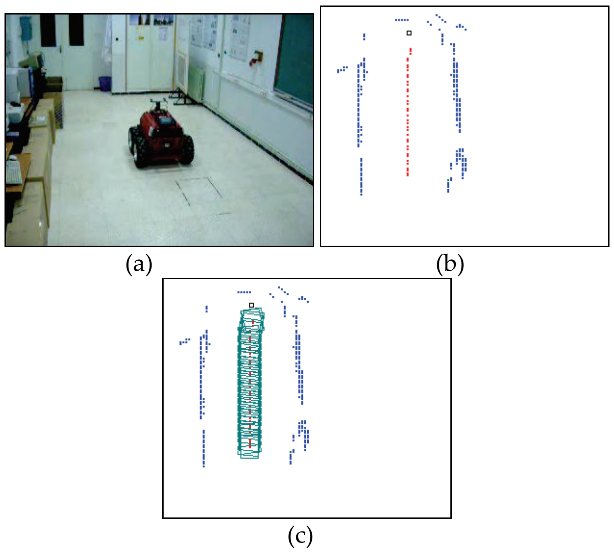


Fig. 11. First experiment: (a) The *ATRV2* robot environment, (b and c) The reproduced histogram grid of the global map with the robot shape and its gravity center trajectory.

The second experiment is similar to the first; the start position and the goal position were taken aligned except those where there are some obstacles which cross the path in front of the robot. The reproduced histogram grid is shown in fig.12.b. When the mobile robot starts moving, it detects obstacles in the front and in the right. The robot makes the correct decision and turns to the left avoiding the detected obstacles successfully and reaches the goal.
The third experiment is a typical experiment to show the performance of the developed *DVFF* method when the mobile robot is trapped in local minima. In this experiment, the start position and the goal position were taken aligned too. The mobile robot starts to move on a straight line. When it is in displacement, it detects obstacles forming a dead end using its onboard ultrasonic sensors. It calls then the function *Case_D** and follows the new processed global path information. The figure 13 shows that the mobile robot avoids perfectly the detected dead end.
In the fourth experiment, the start position and the goal position are aligned to the left of the mobile robot obstructed by an obstacles wall. The mobile robot starts to turn on the left according to the global path information of the cell under the CP$_1$ point. As it moves, it detects obstacles from its frontal ultrasonic sensors. The function *Case_D** makes a correct decision by indicating the right

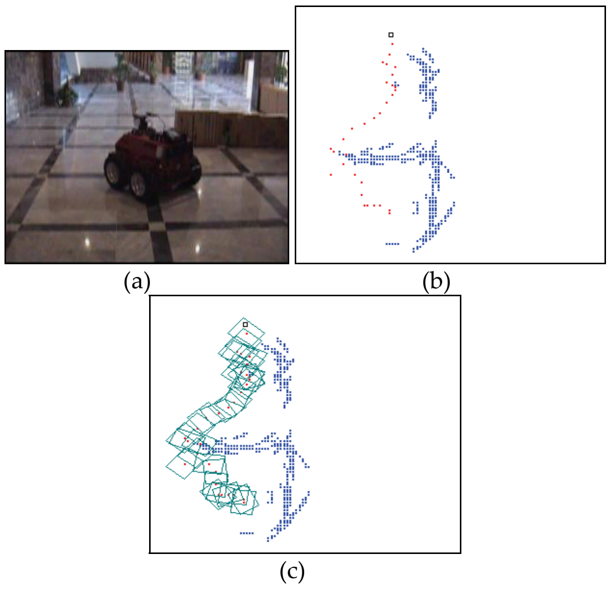


Fig. 12. Second experiment: (a) The *ATRV2* robot environment, (b and c) The reproduced histogram grid of the global map with the robot shape and its gravity center trajectory.

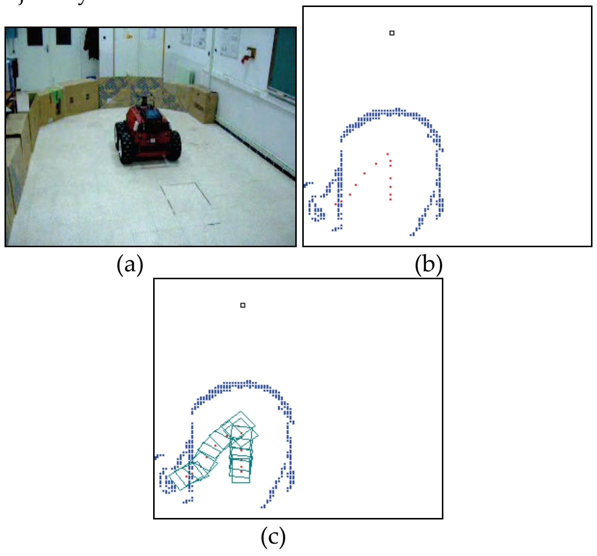


Fig. 13. Third experiment: (a) The *ATRV2* robot environment, (b and c) The reproduced histogram grid of the global map with the robot shape and its gravity center trajectory.

direction to the mobile robot until it reaches the goal. The figure 14 shows that the mobile robot avoids perfectly the wall and reaches the goal represented by a small rectangle.
The last experiment is a typical experimental run using the algorithm developed in (Oualid Djekoune, Karim Achour & Redouane Toumi, 2005). The mobile robot is in the same situation as in the third experiment. The aim of this experiment is to show the mobile robot behavior when it uses only the local information from the onboard sensors for navigation. The figure 15 shows when the mobile robot entered a dead end, it is trapped automatically in local minima.
The choice of the coefficients (α, β, γ) of the equation (6) isn't simple and there isn't any known method to find them efficiently in an automated way. We have chosen them experimentally.

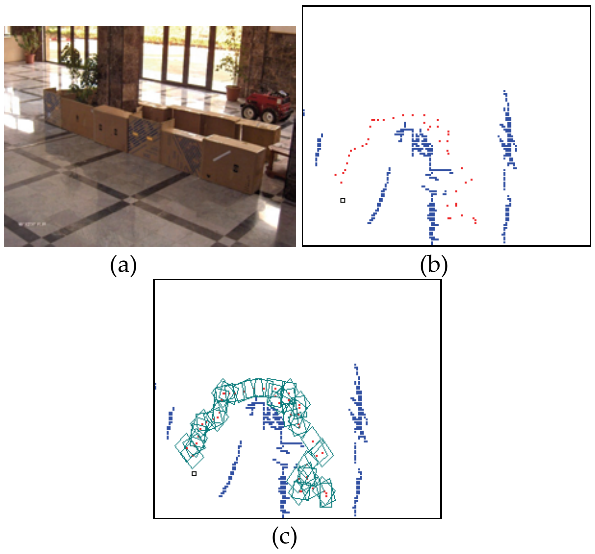


Fig. 14. Fourth experiment: (a) The *ATRV2* robot environment, (b and c) The reproduced histogram grid of the global map with the robot shape and its gravity center trajectory.

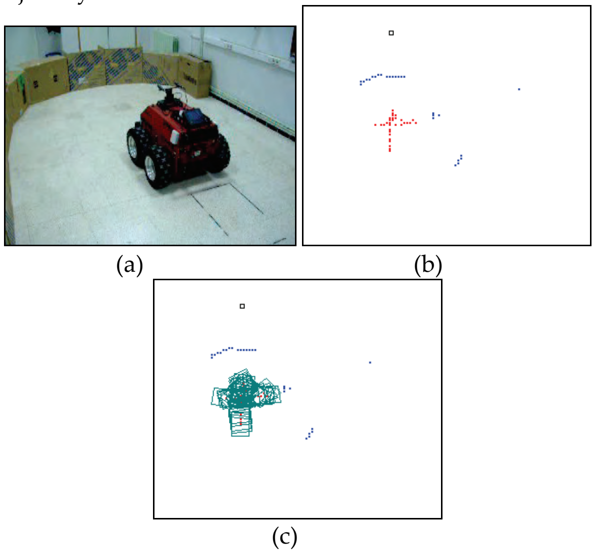


Fig. 15. Fifth experiment: (a) The *ATRV2* robot environment, (b and c) The reproduced histogram grid of the global map with the robot shape and its gravity center trajectory.

At each point along the robot trajectory, a local map of the environment was obtained and integrated into a global map. The global map is represented by a matrix to store information about different cells in the map. The map size is given by the size of the matrix and the area that is to be included in the map. In our case, the max size of the covered area is 10×10 meters and the size of each cell is 10×10 centimetre.

9. Conclusion and Future Work

In this paper, we have solved the problem of both the incompleteness of the environment map and the use of only a reactive navigation method for a mobile robot.
Our approach called *DVFF* based on the idea of combining the global path planning based on *D** algorithm with a real time obstacle avoidance algorithm based on the *VFF* performs a path to reach a goal position in unknown environment. The originality of this approach lies in the use of the *D** algorithm only to generate the global path information from the current robot position in the environment towards the goal. The global path information is the backpointer direction. The mobile robot follows this information if it exists from its position towards the goal. Otherwise, it generates a free collision motion to move around the detected obstacles.
No prior knowledge about the environment is assumed in this approach. Such knowledge can be provided in form of the environment model or be acquired during motion through sensing. The map is built using a modified *HIMM* algorithm based on range data sampled only by onboard ultrasonic sensors, and can work as a good platform for fusion of different kinds of sensor data, as long as other readings from other sensors have sensor models. The updating rule does not use any probability functions and takes into account all the grids inside a sector for each sonar reading.
A satisfactory results have been obtained concerning the problem of mobile robot navigation in unknown environments but some improvements can be brought like the automatic settings of the three coefficients (α, β, γ), taking into account the topological information of the obstacle's position and using information from the onboard stereo pair camera will certainly enhance the mobile robot navigation quality.
In the near future, we would like to use this new approach with developed algorithms such as in (Djekoune, O. & Achour, K., 2004) (Achour, K. & Djekoune, A. O., 2002) in outdoor environments for monitoring for the first time our laboratory site.

10. Acknowledgments

The authors would like to thank all vision team members of the Robotic and Industrial automation Division for useful discussions and comments.

10. References

Achour, K. & Djekoune, A. O. (2002). Localisation and guidance with an embarked camera on a mobile robot. *Advanced Robotics*, Vol. 16, No. 1, 2002, pp. 87-102, ISSN 0169-1864.

Borenstein, J. & Koren, Y. (1990). Real-time obstacle avoidance for fast mobile robots in cluttered environments, Proceeding of the IEEE International Conference on Robotics and Automation, Vol. 1, pp. 572-577, ISBN 0-8186-9061-5, Cincinnati, 13-18 May 1990, OH, USA.

Borenstein, J. & Koren, Y. (1991). The vector field histogram – fast obstacle avoidance for mobile robots, *IEEE Transactions on Robotics and Automation*, Vol. 7, Issue 3, June 1991, pp. 278-288, ISSN 1042-296X.

Borenstein, J. & Koren, Y.(1991). Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Journal of Robotics and Automation*, Vol. 7, No 4, 1991, pp. 535-539, ISSN 0882-4967.

Borenstein Johann & Raschke Ulrich (1991). Real-time Obstacle Avoidance for Non-Point Mobile Robots, Proceedings of the Fourth World Conference on Robotics Research, pp. 2.1–2.9, Pittsburgh, PA, Sept 1991, USA.

Borenstein Johann & Ulrich Iwan (1998). Reliable obstacle avoidance for fast mobile robots, Proceedings of the IEEE Int. Conf. on Robotics and Automation, pp. 1572 - 1577, Leuven, May 16–21, 1998, Belgium.

Borenstein Johann & Ulrich Iwan (2000). VFH*: local obstacle avoidance with look-ahead verification, Procceding of the IEEE Int. Conf. Robotics Automation, pp2505-2511, San Francisco, CA, April. 2000, USA.

Barbehenn, M. & Hutchinson, S. (1995). Efficient search and hierarchical motion planning by dynamically maintaining single-source shortest path trees. *IEEE Transactions on Robotics and Automation*, Vol. 11, No. 2, 1995, pp. 198–214, ISSN 0882-4967.

Brock Oliver & Khatib Oussama (1999). High speed navigation using the global dynamic window approach, Proceedings of the IEEE Int. Conf. on Robotics and Automation, pp.341-346, Detroit, May 1999, Michigan, USA.

Djekoune, O. & Achour, K. (2004). Incremental Hough Transform: An Improved Algorithm for Digital Device Implementation. *Real-Time Imaging*, Vol. 10, Issue 6, Dec. 2004, pp. 351-363, ISSN 1077-2014.

Djekoune Oualid, Achour Karim & Toumi Redouane (2005). Ultrasonic sensing based navigation for mobile robot with obstacles avoidance, Procceding of the IEEE International Computer Systems & Information Technology Conference, IEEE CSIT'05, pp.193-198, Algiers , July 19-21, 2005, algeria.

Djekoune, A. Oualid ; Achour Karim & Toumi Redouane (2007). A new approach for mobile robot navigation. Proceedings of the Conference sur le Génie Electrique CGE'05, Ecole Militaire Polytechnique, Bordj El Bahri, 16-17 avril 2007, Algiers, Algeria.

Ersson, T. & Hu, X. (2001). Path planning and navigation of mobile robots in unknown environments, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Volume 2, Issue, 2001, pp.858 – 864, Maui, oct 2001, Hawaii, USA.

Fox, D.; Burgard, W. & Thrun, S. (1997). The dynamic window approach to collision avoidance, *IEEE Robotics & Automation Magazine*, Vol. 4, Issue 1, Mar 1997, pp. 23-33, ISSN 1070-9932.

Ferguson Dave & Stentz Anthony (2005). The Delayed D* Algorithm for efficient path replaning, Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2045-2050, Barcelona, April, 2005, Spain.

Gourley Chris & Mohan M. Trivedi (1994). Sensor Based Obstacle Avoidance and Mapping for Fast Mobile Robots, Proceedings of the International Conference on Robotics and Automation, ICRA'94, pp.1306-1311, San Diego, CA, May 1994, USA.

Huiming, Y.; Chia-Jung, C.; Tong, S. & Qiang, B. (2001). Hybrid evolutionary motion planning using follow boundary repair for mobile robots. *Journal of Systems Architecture*, Vol. 47, 2001, pp. 635–647, ISSN 1383-7621.

Ko, N.Y. & Simmons, R. (1998). The lane curvature method for local obstacle avoidance, Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'98, pp. 1615-1621, Victoria, October 1998, Canada.

Koenig, S. & Likhachev, M. (2002). Improved fast replanning for robot navigation in unknown terrain, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Volume 1, pp. 968- 975, ISBN 0-7803-7273-5, Washington, DC, May 11-15, 2002, USA.

Khatib, O. (1986). Real time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, Vol.5, Issue 1, Spring 1986, pp. 90-98, ISSN 0278-3649.

Lumelsky, V. J. & Stepanov, A. A. (1986). Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment0. *IEEE Transactions on Automatic Control*, Vol. AC- 31, No. 11, November 1986, pp. 1058- 1063, ISSN 0018-9286.

Minguez, J. & Montano, L.(2000). Nearness diagram navigation (ND): A new real time collision avoidance approach, Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'00, pp.2094-2101, 2000, Takamatsu, Japan.

Moravec, H. P. (1988). Sensor Fusion in Certainty Grids for Mobile Robots. *AI Magazine*, Summer 1988, pp. 61-74.

Macek, K.; Petrovic, I. & Ivanijko, E. (2003). An approach to motion planning of indoor mobile robots, Proceedings of the IEEE International Conference on Industrial Technology, ICIT'03, pp. 969-973, 2003, Maribor, December 10 - 12, 2003, Slovenia.

Podsedkowski, L.; Nowakowski, J.; Idzikowski, M. & Vizvary, I. (2001). A new solution for path planning in partially known or unknown environments for nonholonomic mobile robots. *Robotics and Autonomous Systems*, Vol. 34, 2001, pp. 145–152,ISSN 0921-8890.

Ramalingam, G. & Reps, T. (1996). An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms*, Vol. 21, 1996, pp. 267–305, ISSN 0196-6774 .

Stenz, A. (1994). Optimal and efficient path planning for partially-known environments, Proceedings of IEEE International Conference on Robotics and Automation, ICRA, pp.3310-3317, San Diego, May 1994, CA, USA.

Stentz, A. (1995). The Focussed D* Algorithm for Real-Time Replanning, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1652-1659, Montréal, August 20-25, 1995, Québec, Canada.

Simmons, R. (1996). The curvature velocity method for local obstacle avoidance, Proceeding of the IEEE International Conference on Robotics and Automation, ICRA'96, pp. 3375–3382, ISBN 0-7803-2988-0, Minneapolis MN, 22-28 April 1996, USA.

Seder, M.; Macek, K.; Petrovic, I. (2005). An integrated approach to real time mobile robot control in partially known indoor environments. Proceeding of the 31st Annual Conference of the IEEE Industrial Electronics Society, pp.1785-1790, Raleigh, Nov. 2005, North Carolina, USA.