ESCAPE LANES NAVIGATOR
To Control "RAOUL" Autonomous Mobile Robot

Nicolas Morette, Cyril Novales and Laurence Josserand
Laboratoire Vision & Robotique,Orleans Universiy,63 avenue de Lattres de Tassigny,18020 Bourges cedex, France

Keywords: Robotics, Mobile robots, Autonomous robot, Navigation, Escape lanes.

Abstract: This paper presents a navigation method to control autonomous mobile robots : the escapes lanes, applied to RAOUL mobile system. First, the formalism is introduced to model an automated system, and then is applied to a mobile robot. Then, this formalism is used to describe the escape lanes navigation method, and is applied to our RAOUL mobile robot. Finally, implementation results simulations validate the concept.

1 INTRODUCTION

In the autonomous robotics field, the navigator is defined as a module whose purpose is to find a relevant trajectory for a robot within its local surrounding area. The navigator has to deal with various robot constraints (such as kinematic constraints or saturations), and has to find a trajectory which avoid collisions with static or mobile obstacles. The trajectory proposed by the navigator respecting these constraints is followed by the robot using the pilot and servoings modules.

There are various methods to generate a path for an autonomous mobile robot in an unknown environment. Artificial potential fields methods (Agirrebeitia, 2005) generate a path along the weakest potential gradient, in a potential fields map where obstacles are associated with strong potentials. The neural networks methods define a succession of nodes in a neural network map of the robot environment (Lebedev, 2005). Fuzzy Logic methods define a set of logic rules and can drive a robot to move safely in its environment (Xu, 1999).

However, according to our definition, a navigation method is a complement to this kind of approach (called path planning methods), whose purpose is to find a trajectory in order to follow a path generated by a path planning method under local constraints. Moreover, a navigation method has to generate smooth trajectories.

Two kinds of trajectory generation methods are identified: those using an inverse model of the robot, and those using a direct model. Among the first ones, the flat output method controls the robot in order to follow a calculated trajectory (Fraisse, 2002). (Munoz, 1994) have parametered B-splines methods to determine a time parametered trajectory for the robot. Among the second ones (Belker, 2002) used an hybrid neural networks method to project acceptable trajectories for the robot in a local neural network map. In this paper, another direct model method based on escape lanes (Novales, 1994) is developed to project acceptable trajectories for the robot in its local environment.

2 MODEL OF THE ROBOT

Lots of mobile robot navigation methods are based on inverse model: the robot desired trajectory is given by the inverse model, which delivers the articular set points for the servoings. Due to the mechanical design, sometimes there is no solution (non-holonomy). Moreover, if the robot evolves in a constrained environment, cartesian constraints must be expressed in the robot articular space.

We have chosen to use a direct model of the mobile robot to control it. There is always a solution, and the environmental constraints can be expressed directly in the cartesian space. As a consequence, we need to use a global model of the robot and of its environment, which allows to project all the admissible trajectories in a near future (a time horizon of few seconds).

Morette N., Novales C. and Josserand L. (2007).
ESCAPE LANES NAVIGATOR - To Control "RAOUL" Autonomous Mobile Robot.
In *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics*, pages 45-52
DOI: 10.5220/0001625600450052
Copyright © SciTePress

2.1 Formalism of an Automated System

Based on the formalism developed by Sontag (Sontag, 1990), a mobile robot can be modeled with spaces and functions defined as follows (Figure 1):

$$\Sigma = (t, \chi, U, \varphi, \Psi, h) \tag{1}$$

where t is the time space,

U is the input space of the system; u are the vectors of this input space,

χ is the state space of the system; X are the vectors of this state space,

Ψ is the output space of the system; Y are the vectors of this output space,

φ is the state function which gives a current state vector X_a (at time t_a) knowing the initial time t_0, the current time t_a, the initial state X_0 and the input function ω (defined later).

h is the output function which gives the current output vector Y_a (at time t_a) knowing the current time t_a and state X_a.

The input function ω associates an input vector u_a to the current time t_a ($\omega(t_a) = u_a$).

The transfer function λ associates the current output vector Y_a to the current input vector U_a at time t_a and the input function ω (usually it is $\varphi \circ h$).

Sontag defines simplified functions $\xi(t)$ and $\lambda(t)$ associating either a state vector or an output vector directly from the current time.

$\xi(t)$, the simplified state function, makes it possible to define state X_a of the system time t_a: $\xi(t_a) = X_a$. Note that ξ is the image of ω by φ in the state space: $\xi(t_a) = \varphi(t_0, t_a, X_0, \omega)$

$\bar{\lambda}$ (t) is the simplified exit function which gives the exit vector Y_a of the system at time t_a: $\bar{\lambda}(t_a) = Y_a$. Note that $\bar{\lambda}$ is the image of ξ by h in the space of exit ψ: $\bar{\lambda}(t_a) = h \circ \xi(t_a)$.

2.2 Spaces Specification

In the case of a mobile robot, the contents of the vectors associated with each space are specified, in order to introduce our navigation method in section 3.

In the input space U of the robot, vectors contains set points of the actuator servoings input, i.e. the articular velocities of the robot: $u = (\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n)^T$.

In the state space χ of the mobile robot, the state vectors are defined as kinematics values, i.e. the curvilinear velocities and the rotation velocity of the mobile robot: $X = (\dot{s}, \dot{\theta})^T$.

The robot output space Ψ is the configuration space (Lozano-Perez, 1983), where output vectors are defined as the coordinates/orientation of the robot: $Y = (x, y, \theta)^T$.

This implies the output function becomes a differential function and does not depend only on the current state. The new output function L which gives the current output vector Y_a at the current time t_a depends on the initial time t_0, on the initial output Y_0 and on the simplified state function ξ.

Our mobile robot is then defined by $\Sigma = (t, \chi, U, \varphi, \Psi, L)$.

2.3 Definition of a Trajectory

A trajectory Γ of a given system Σ defined on an interval of time $[t_0, t_0 + \tau]$ is composed by the simplified state function ξ and the function of entry ω defined on the same interval: $\Gamma = (\xi, \omega)$

nb: the simplified exit function $\bar{L}$ (t) is the image of the trajectory Γ(t) in the output space. Typically, it is the "trace on the floor" of the trajectory of the robot.

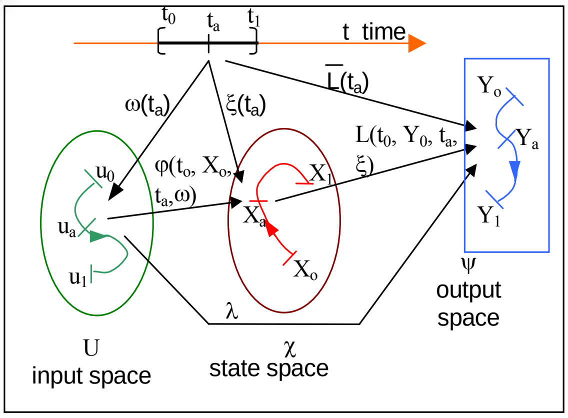

Figure 1: Automated system $\Sigma = (t, \chi, U, \varphi, \Psi, L)$.

3 ESCAPE LANES FORMALISM

Similarly to animal strategy, our navigation is based on direct models. When a mobile robot is considered in a given state, we can project in a few seconds horizon all trajectories that it can perform. These trajectories can be blended with the environment and with the motion goal to select the most appropriate trajectory. These selected trajectories become the new set points of the mobile robot. In order to give the ability to the robot to react in real time to its environment, the process is repeated periodically.

The different steps of the escape lanes method are to:

- generate all acceptable trajectories Γ to be performed by the robot (called escape lanes) on a temporal horizon τ,
- eliminate the blocked escape lanes (e.g. intersect or pass too close to obstacles),
- choose a free escape lane for the robot, using a selection criterion.

The strong point of this method is that it uses only the robot direct model, the inverse function of φ does not need to be defined.

The whole operation is reiterated periodically and is represented on Figure 2.

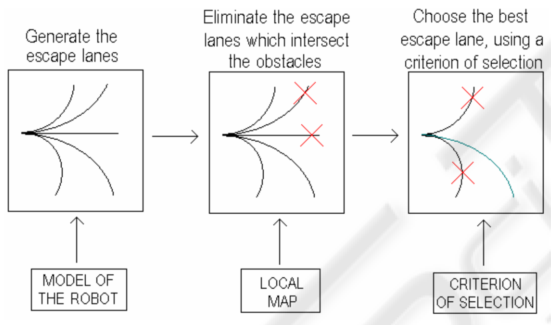
Generate the escape lanes
Eliminate the escape lanes which intersect the obstacles
Choose the best escape lane, using a criterion of selection
MODEL OF THE ROBOT
LOCAL MAP
CRITERION OF SELECTION

Figure 2: Escape lanes principle.

3.1 Acceptable Trajectories by the Robot and Escape Lanes

An input function ω is acceptable by the robot on an interval $[t_0, t_0+ \tau]$ if it respects the following constraints:

-constraints corresponding to the direct kinematics model of the robot, *i.e.* its possibilities of movements (kinematics constraints),

-constraints of actuators saturation,

-constraints on the dynamics of the actuators (maximum accelerations, saturations…)

$\Omega[t_0, t_0+ \tau]$ is called the entire set of the acceptable input functions on the interval $[t_0, t_0+ \tau]$:

$$\Omega[t_0,t_0+\tau]=\{\omega \in U[t_0,t_0+\tau] / \omega \text{ acceptable } \} \qquad (2)$$

$\Lambda[t_0, t_0+ \tau]$ is called the set of the acceptable trajectories by the robot on the interval $[t_0, t_0+ \tau]$. It corresponds to the set of trajectories associated with the input function $\omega \in \Omega[t_0, t_0+ \tau]$. $\Lambda[t_0, t_0+ \tau]$ is also called the escape lanes of the robot at time t_0 and on the temporal horizon τ.

$$\Lambda[t_0,t_0+\tau]=\{\Gamma=(\xi,\omega) / \omega \in \Omega[t_0,t_0+\tau] \} \qquad (3)$$

3.2 Elimination of the Blocked Escape Lanes

The following stage consists in comparing the image of each trajectory $\Gamma \in \Lambda$ in the output space Ψ to the obstacles map called Θ. This map can be displayed in several forms, but must imperatively present the obstacles position with respect to the robot. Using the elimination criterion C_{el}, the ensemble of the free escape lanes Λ_L is determinated by:

$$\Lambda_L=\{\Gamma=(\xi,\omega) / C_{el}(\lambda_\Gamma,\Theta)=0, \Gamma \in \Lambda[t_0,t_0+\tau] \} \qquad (4)$$

Escape lanes which intersect the obstacles or pass too close to them are eliminated ($C_{el} \neq 0$), according to C_{el}. The remaining escape lanes constitute the free escape lanes set Λ_L.

By association, the ensemble of the free acceptable input functions Ω_L can be defined as the entire set of the input functions ω which correspond to the free escape lanes Λ_L.

$$\Omega_L=\{\omega \in \Omega / , \Gamma \in \Lambda_L \} \qquad (5)$$

3.3 Selection of the Best Free Escape Lane

The last stage consists in determining the best escape lane among Λ_L to reach a target ε provided by the higher control level (*i.e.* the path planner).

To select the best free escape lane a criterion C_{choice}, is applied to quantify the relevance of each free escape lane to achieve this goal. The selected escape lane is associated to the optimal value of the criterion, and is called Γ_{chosen}. The corresponding function of entry is called ω_{chosen}, and is sent to the lower control level (*i.e.* the pilot).

$$\Gamma_{chosen}=\{\Gamma=(\xi,\omega) / C_{choice}(\Gamma,\varepsilon) \text{ optimal}, \Gamma \in \Lambda_L \} \qquad (6)$$

$$\omega_{chosen}=\{\omega \in \Omega_L / \Gamma=\Gamma_{chosen\,L} \} \qquad (7)$$

3.4 Discussion on Criteria

The choice of the C_{choice} and C_{el} criteria depends on the application to be carried out by the robot. For an exploration mission, when the purpose of the robot is to chart the surrounding area, a severe criterion of elimination C_{el} has to be established to make sure that the robot passes at a safe distance away from obstacles (the obstacles positions are known, but their shapes and volumes are not). A criterion of selection C_{choice} that supports the most efficient trajectories in term of energy may be used.

On the other hand, when considering missions of an industrial type, the robot velocity is favored, and thus the C_{choice} criterion must support the fastest trajectories (at the expense of the energy consumption). Thus, these criteria must be adapted to the situation (mission and kind of environment).

The input function corresponding to the selected trajectory (ω_{chosen}) is actually applied to the robot. Indeed the complete operation (from the generation of escape lanes of the robot to the selected selection of Γ_{chosen} and ω_{chosen}), is performed periodically, and with a period T_e about ten times smaller τ. Therefore only a short portion of the Γ_{chosen} trajectory is actually followed by the robot; a new one is proposed every T_e.

4 APPLICATION TO RAOUL MOBILE ROBOT

4.1 Presentation of RAOUL

RAOUL (figure 3) is an autonomous mobile robot, able to run in an unknown environment, finding a trajectory on its own to avoid collision with static or mobile obstacles. Raoul is built on a Robuter platform (Robosoft) with an additional computer (PC/RTAI-Linux) and exteroceptive sensors: two Sick telemeters laser (a front one and a rear one) and one laser goniometer.

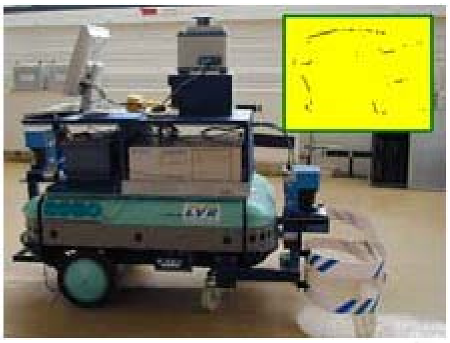

Figure 3: The RAOUL robot.

The control architecture (figure 4) is a multi-level architecture developed by the laboratory of vision and robotic of the University of Orleans (Mourioux, 2006). Each level corresponds to a perception/action loop. Low levels correspond to fast reaction behaviors and high levels to "intelligent" behaviors.

Level 0 represents the Articulated Mechanical System (AMS). Level 1 of the architecture corresponds to the servoings loop using proprioceptive sensor informations. On level 2, the "pilot" module performs emergency reactive decisions, using data from the two laser range telemeters. The third level, the "navigator", is in charge of finding a local trajectory for the robot, using a local map module (Canou, 2004). On the upper level, the "path planner" has the mission to find a global path for the robot.

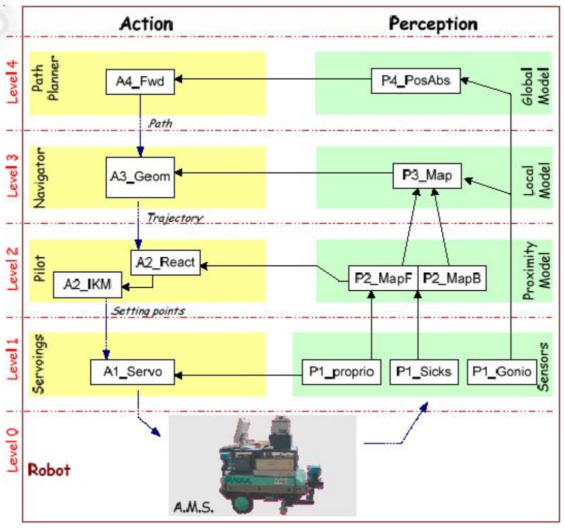

Figure 4: the control architecture.

The part developed in this work concern only the navigation level.

4.2 Acceptable Trajectories

First the constraints of our system are determined to define the ensemble of the functions of acceptable entry Ω on a temporal horizon τ.

RAOUL is a mobile robot with differential wheels; we assume that the motion is done without slipless on the ground. According to the trigonometrical direction for w_1 and w_2 (the rotation velocity of the two driving wheels) we obtain:

$$\dot{s} = \frac{s_1 + s_2}{2} = \frac{w_2 R - w_1 R}{2} \tag{8}$$

$$\dot{\theta} = \frac{s_1 - s_2}{2L} = \frac{w_2 R + w_1 R}{2L} \tag{9}$$

Where s corresponds to the curvilinear distance traversed by point C, and θ the orientation of the robot.

The dynamic constraints and saturations depend on the robot itself and correspond to $w_{\min}$, w_{max}, $\dot{w}_{\min}$, $\dot{w}_{\max}$.

It is then necessary to choose a family (or a number finished families) of input functions, that must be selected according to the robot capacities. For example, a car-like robot is not controlled the same way as a robot with differential wheels.

Linear functions are used for RAOUL robot and are represented on Figure 5.

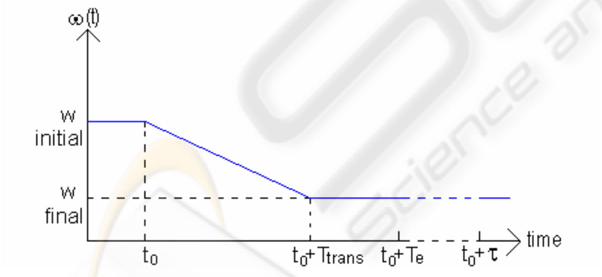

Figure 5: Function of entry.

This family of input functions can be expressed in the following way:

$$w(t) = w_{initial} + \frac{(w_{final} - w_{initial}) \times (t - t_0)}{T_{trans}} \quad for \quad t_0 < t < t_0 + T_{trans} \tag{10}$$

Under the constraints:

$$\begin{aligned} &w_{\min} \le w_{initial} \le w_{\max} \\ &w_{\min} \le w_{final} \le w_{\max} \\ &\dot{w}_{\min} \le \frac{(w_{final} - w_{initial})}{T_{trans}} \le \dot{w}_{\max} \end{aligned} \tag{11}$$

From this family of acceptable input functions, we obtain a family of acceptable trajectories Λ_{limited} :

$$\Lambda_{LIMITED} = \{\Gamma = (\xi, w_{familly}) \,/\, w_{familly} \in \Omega[t_0, t_0 + \tau]\} \tag{12}$$

With:

$$\begin{aligned} &w_{familly} = (w_1(t), w_2(t))^T \\ &\xi = (\dot{s}, \dot{\theta}) \end{aligned} \tag{13}$$

Where:

$$\begin{aligned} &\dot{s} = \frac{R}{2} \cdot (w_2(t) - w_1(t)) \\ &\dot{s} = \frac{R}{2} \cdot \left\{ (w_{2i} - w_{1i}) + [(w_{2f} - w_{2i}) - (w_{1f} - w_{1i}] \cdot \frac{t - t_0}{T_{trans}} \right\} \end{aligned} \tag{14}$$

$$\begin{aligned} &\dot{\theta} = \frac{R}{2} \cdot (w_1(t) + w_2(t)) \\ &\dot{\theta} = \frac{R}{2} \cdot \left\{ (w_{2i} + w_{1i}) + [(w_{2f} - w_{2i}) + (w_{1f} - w_{1i}] \cdot \frac{t - t_0}{T_{trans}} \right\} \end{aligned} \tag{15}$$

nb: these trajectories are projected starting from the X_0 state and of the Y_0 output of the robot at time t_0.

4.3 Elimination of the Blocked Escape Lanes

To eliminate the blocked escape lanes a comparison criterion between Λ_{limited} and the map of the obstacles Θ is proposed.

The local model of the robot environment made by (Canou, 2004) is used, built in-line using 2 laser-range finders. The obstacles map Θ is composed as a set of segments of known two ends coordinates. These segments represent the obstacles perimeter in the local environment of the robot, projected within the moving plane of the robot.

To determine if an escape lane of Λ_{limited} is free or blocked, the C_{el} elimination criterion is used between its image λ_i in Ψ and each obstacle segment of Θ. C_{el} (λ_i, Θ) = 0 if:

$$dist\left\{\lambda_i(t) / sgmt_j \right\} - (L + margin) > 0 \ \forall \ j. \tag{16}$$

L: maximum distance between the periphery of the robot and the center C of the axis connecting the two driving wheels of the robot; L is also called width of the robot.

dist{$\lambda_i(t)$ / sgmt$_j$} is the minimal distance between λ_i points and the points belonging to segment $_j$.

Escape lanes that do not verify C_{el} (λ_Γ, Θ) = 0 are eliminated. The ensemble of the remaining escape lanes is $\Lambda_{L\,/\,limited}$.

4.4 Selection of the Best Free Escape Lane

Finally a criterion is given to choose the escape lane to be kept as set point for the robot. The choice of this criterion takes into account the distance and the orientation of the robot at the end of the trajectory, compared to its target ε (provided by the path-planner).

The criterion is given by:

$$C_{choice}(\Gamma_i, \varepsilon) = \sqrt{(x_i(t_0+\tau) - x_\varepsilon)^2 + (y_i(t_0+\tau) - y_\varepsilon)^2} \times fact \tag{17}$$

$$fact = 1 + k_\theta \times \left| \theta_i(t_0+\tau) - \theta_{target/robot} \right| \tag{18}$$

k_θ is a positive real fixed empirically by carrying out simulations and experimentations.

This criterion computes the cartesian distance from the robot with ε by balancing it with its orientation compared to ε.

The $\Lambda_{L\,/\,limited}$ escape lane which minimizes this criterion is chosen and called Γ_{chosen}. The associated input function, ω_{chosen}, is applied to the robot.

To improve the relevance of the elimination criterion C_{el}, the margin value in the formula (17) may be adjusted according to the orientation of the robot with respect to the obstacle (C-obstacles (Lozano, 1983)).

5 IMPLEMENTATION

Matlab was used to carry out simulations and the implementation on the robot was made in C language on Linux RTAI system.

5.1 Discretization of the Input Space

In our case the input space is discretized on 5 by 5 mode, *i.e.* when the generation of possible trajectories is carried out, 5x5 = 25 trajectories are generated corresponding to the 25 possible input functions ; each one moving towards one of the 25 possible couples (w_1, w_2) reached at the end of T_{trans}.

The more the space of entry is discretized in a great number, the greater trajectories number is possible for our robot at time t_0. As a consequence the computation time is higher. It is thus necessary to find a compromise.

5.2 Generation Off-Line or On-Line of $\Lambda_{Limited}$

Experimentally, there are two possibilities to carry out the generation of the possible $\Lambda_{limited}$. Thos generation can be performed on line, *i.e.* while the robot is moving and just before their application, or to do it partially off line.

When fully done on line, 5x5=25 trajectories have to be generated for the robot (or n² if Ω is discretized by n on n). That is multiplied by the number of points on each trajectory, *i.e.* τ divided by the step of calculation over time. In simulations, a temporal horizon τ=3s is used for a step of 0.05s, *i.e.* a total of 60 points per trajectory and thus 1500 points overall.

Another solution consists in carrying out the generation of $\Lambda_{limited}$ off line. However, it is then necessary to generate 5x5 x 5x5 = 625 trajectories. The trajectories to join the 25 discretized final couples (w_1, w_2) have to be computed, with the 25 possible initial couples as a starting point. It is then necessary to store all these trajectories during on-line navigation; the 25 trajectories corresponding to the initial state of the robot are projected, using the following transform matrix:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t+t_0} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t_0} + \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t/\Gamma_{chosen}} \tag{19}$$

This solution is used when the robot does not have the necessary power to perform all the calculations on line. The totality of the generated trajectories is represented on figure 6.

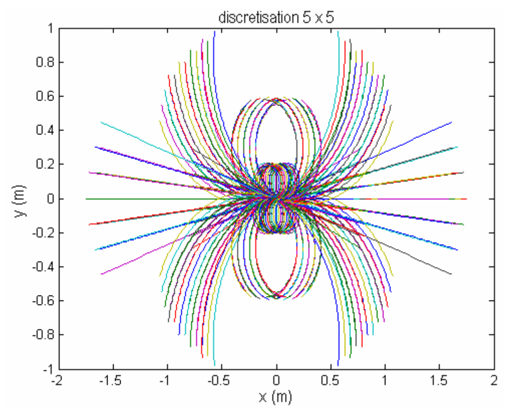

Figure 6: Possible trajectories of the robot stored in memory.

5.3 Implementation

Experimentally, we are constrained to perform the navigation calculation one period T_e ahead of time. Indeed, calculations for the period $[t_0, t_0+T_e]$ as from time t_0 since it takes a certain time for calculation and since the robot continues to move forward during this time.

The navigation calculation for the period $[t_0, t_0+T_e]$ must thus be carried out during the period $[t_0-T_e, t_0]$. To know the robot output Y at t_0, we carry out the approximation which the robot will have followed exactly the escape lane chosen over the period $[t_0-T_e, t_0]$, starting from its output Y at the moment $t_0- T_e$.

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t_0} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t_0-T_e} + \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}_{t_0-T_e} \bullet \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{\substack{T_e / \Gamma_{chosen} \\ on\,[t_0-T_e, t_0]}} \quad (20)$$

6 RESULTS

6.1 Simulation

The simulation program that we have developed is composed of three parts. The first part calculates and stores in memory the possible movements of the robot (off-line generation of Λ_{limited}).

The second creates a chart of a virtual environment in which the robot will navigate. This environment consists in obstacles viewed as segments (these segments represent the periphery of the obstacles in the navigation plan).

The last part corresponds to navigation itself, and includes the projection stages of the robot escape lanes, the elimination of the blocked escape lanes and the choice of the most suitable escape lane. This part is performed until the virtual robot reaches a desired localization (beyond a certain number of loops the program stops). Finally the program traces the path that the robot carries out on the chart, and gives the inputs list sent to the robot with each iteration.

6.2 Results

Figure 7 shows a simulation representing displacements of the robot using a differential wheels type model. The robot starts at point coordinates (-8, -8), and must go to (-5, 4).

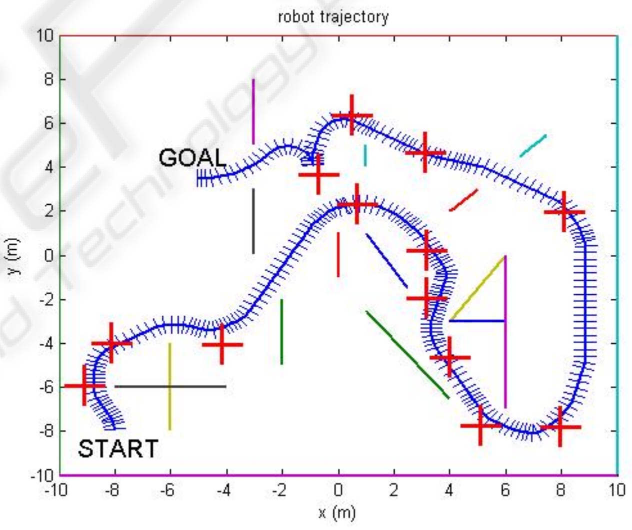

Figure 7: Simulation of obtained trajectory.

The trace of the displacements achieved by the center of the robot is represented in blue, and each new iteration of the program of navigation is symbolized by a small feature perpendicular to the trace. The red crosses represent the points of passage provided to the navigator.

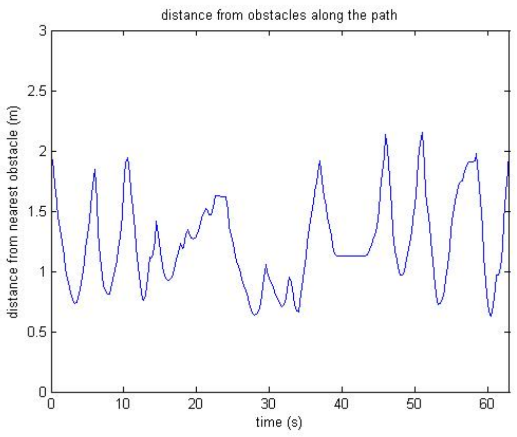

Figure 8: distance robot/obstacle.

The distance between the robot and its nearest obstacle along the path is represented on figure 8. We can note that the robot never enters in collision with the obstacles because the distance from obstacles is larger than the robot width (50cm). Moreover in the event of appearance of an unexpected obstacle, the pilot (*i.e.* the lower level of control) has the capacity to perform reflex decisions, of absolute priority, to avoid this kind of obstacle in emergency. We can note the great flexibility of the generated trajectories.

7 CONCLUSION

This study showed that escape lanes is an efficient navigation method, able to generate fluent moves for RAOUL autonomous mobile robot. Its strong point is that it only uses the direct kinematics model of the robot, ensuring that the robot is actually able to perform the desired moves. Indeed it's more difficult to transpose constraints in the input space of the robot using an inverse model.

However it's only a navigation method, meaning that it must work cooperatively with a path planner module, which gives a global path to follow to the navigator. The purpose of the navigator is to follow this path as close as possible, under the local constraints of the environment the robot evolves in.

The next step is to implement this method with a planner on a Cycab robot, using a GPS for the path planning module. Cycab is a car-like robot, with only one mobility degree. This will provide a stronger non holonomy constraint in the implementation of the proposed navigation method.

REFERENCES

Agirrebeitia, J., Aviles, R., de BUSTOS, I. F., Ajuria, G., June 2005, *A new APF strategy for path planning in environments with obstacles*, Mechanism and Machine Theory, Volume 40, Issue 6, , Pages 645-658

Belker, T., Schulz, D., *Intelligent Robots and System, 2002, Local Action Planning for Mobile Robot Collision Avoidance*, IEEE/RSJ International Conference on Volume 1, 30 Sept.-5 Oct. 2002 Page(s):601 - 606 vol.1

Canou, J., Mourioux, G., Novales, C. Poisson, G., April 26 – May 1st 2004 , *A local map building process for a reactive navigation of a mobile robot*, Proceedings of IEEE International Conference on Robotics and Automation, pp4839-4844, New Orleans, USA

Fraisse, P., Gil, A. P., Zapata, R., Perruquetti, W., Divoux, T., 2002, *Stratégie de commande collaborative réactive pour des réseaux de robots.*

Lebedev, D. V., April 2005. *The dynamic wave expansion neural network model for robot motion planning in time-varying environments, Neural Networks*, Volume 18, Issue 3, Pages 267-285.

Lozano-Perez, T., 1983, *Spatial Planning: a Configuration Space Approach, IEEE Transaction and computers,* vol. C-32, no. 2, Fev. 1983.

Novales, C., 1994, *Navigation Locale par Lignes de Fuite, rapport de thèse : Pilotage par actions réflexes et navigation locale de robots mobiles rapides*, chapitre IV, soutenue le 20 octobre 1994, Pages 87 à 107

Novales, C., Mourioux, G., Poisson, G., April 6,7 2006 *, A multi-level architecture controlling robots from autonomy to teleoperation*, First National Workshop on Control Architectures of Robots – Montpellier

Munoz, V., Ollero, A., Prado, M., Simon, A., 1994 IEEE International Conference on 8-13 May 1994, *Mobile robot trajectory planning with dynamic and kinematic constraints, Robotics and Automation*, 1994. Proceedings, Page(s):2802 - 2807 vol.4

Sontag, E. D., 1990. *Mathematical control Theory – Deterministic Finite Dimensional Systems*, ED-Spronger-Velag New-York 1990.

Xu, W. L., August 1999, *A virtual target approach for resolving the limit cycle problem in navigation of a fuzzy behavior-based mobile robot*, Institue of Technology and Institute of Technology and Engineering, College of Sciences, Massey University, Palmerston North, New Zealand ; Received 22 June 1998 ; accepted 20 August 1999