# Evaluation de la sûreté de techniques de navigation autonome

Stephane Laforet

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

---

EXAMEN PROBATOIRE
EN INFORMATIQUE

Présenté par

**Stéphane LAFORET**

---

# EVALUATION DE LA SURETE DE TECHNIQUES DE NAVIGATION AUTONOMES

---

Soutenu le 31 mars 2006, à Grenoble, devant le jury:

| | |
|---|---|
| Présidente: | Mme Véronique DONZEAU-GOUGE (CNAM Paris) |
| Examinateurs: | M. Eric GRESSIER (CNAM Paris) |
| | M. Jean-Pierre GIRAUDIN (CNAM, UPMF Grenoble) |
| | M. André PLISSON (CNAM Grenoble) |
| Tuteur: | M. Thierry FRAICHARD (INRIA Rhône Alpes) |

## REMERCIEMENTS

*Je tiens à remercier tout d'abord mon tuteur M. Thierry FRAICHARD, chargé de recherche à l'INRIA, pour m'avoir guidé tout au long de la réalisation de ce travail. J'ai apprécié sa disponibilité et le partage de ses connaissances en robotique. Plus généralement, je le remercie pour l'intérêt accordé à ce projet.*

*Mes remerciements vont également à M. Giraudin Jean-Pierre, Professeur en Informatique au centre CNAM de Grenoble, pour sa confiance exprimée par l'attribution de ce sujet.*

*Je souhaite également remercier Mme Donzeau-Gouge Véronique, Présidente du Jury, ainsi que tous les membres du jury, pour s'être intéressés à ce rapport de probatoire informatique.*

*Je souhaite enfin remercier mes proches et l'association des ingénieurs CNAM (AIPST) pour leur soutient, leurs conseils, et leur aide.*

# Sommaire

# Table des figures

## Introduction

Afin de permettre à un robot mobile de se déplacer sans collision vers un but, les recherches en robotique ont donné naissance aux méthodes de navigation autonome. Elles se divisent en 2 familles principales: les méthodes délibératives et les méthodes réactives. Les méthodes délibératives permettent de calculer au préalable un chemin ou un ensemble de trajectoire en se basant sur une connaissance a priori de l'environnement. Les méthodes réactives utilisent l'environnement perçu afin de générer un mouvement à exécuter sur un pas de temps, à la suite duquel un autre mouvement est généré et ainsi de suite. Ces mouvements sont appelés "mouvements partiels".

L'inconvénient majeur des méthodes délibératives est leur incompatibilité avec des contraintes de calculs en temps-réel, rendant difficile leur utilisation en environnement dynamique. Quant aux méthodes réactives, leur inconvénient principal est celui des minimums locaux [1] : d'un point de vu local, une direction peut sembler la plus judicieuse pour atteindre le but alors qu'en réalité, elle conduit à un cul-de-sac (figure 1).

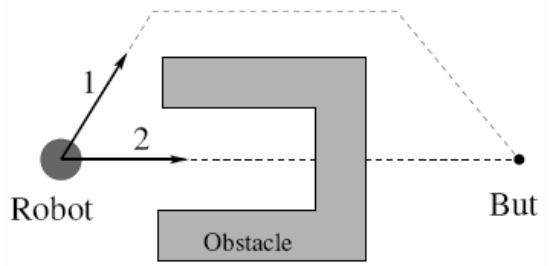


*figure 1 : la direction 2 semble localement plus judicieuse que 1 pour atteindre directement le but*

Ainsi, pour pallier aux problèmes inhérents à ces 2 familles, un 3ème type d'approche a vu le jour, il s'agit de la famille des méthodes hybrides. Elles sont généralement composées d'un algorithme réactif dont les décisions se font en connaissance d'un chemin planifié par une fonction délibérative.

Quelle que soit la famille à laquelle appartienne ces méthodes, elles doivent satisfaire 3 contraintes : la **convergence**, la **faisabilité** et la **sûreté**.

La convergence définit la capacité d'un robot mobile à atteindre un but précis après s'être déplacé. La faisabilité assure que le robot considéré est capable d'effectuer les mouvements calculés. La sûreté doit permettre d'accomplir un déplacement en l'absence totale de collision.

Les 5 approches de type hybride étudiées dans ce rapport sont :
- *Vector Field Histogram** (**VFH***) [2],
- *Global Dynamic Window Approach* (**GDWA**) [3],
- *Global Nearness Diagram* (**GND**) [4],
- *Elastic Band + Dynamic Windows Approach* (**EB+DWA**) [5],
- *Velocity Obstacle* (**VO**) [6].

La faisabilité est une contrainte importante qui est généralement satisfaite. Ce qui n'est pas toujours le cas pour la convergence et la sûreté. Or vis-à-vis de l'intégrité du système et de son environnement, seule la sûreté ne peut pas être minimisé aux vues des applications dans lesquelles on souhaite faire intervenir des robots : transport, robot de service, chaise roulante autonome, robot guide etc.

Pour évaluer la sûreté d'une méthode de navigation autonome dans un environnement dynamique, ce rapport va s'appuyer sur un critère objectif : celui d'état de collision inévitable [7] (ECI ou ICS[1]). En effet, considérer la sûreté par le seul fait de ne pas se retrouver dans la zone occupée par un obstacle n'est pas suffisant. La prise en compte de la dynamique du robot et de la trajectoire de l'obstacle implique l'existence d'états où le robot entrera en collision quoi qu'il fasse. Le concept d'ECI définit ainsi l'état pour lequel une collision survient quelle que soit la trajectoire future suivie.

La 1ère partie de ce rapport va s'attacher à décrire les 5 méthodes de navigation autonome proposées afin d'être en mesure de les évaluer dans la 2nd partie. Cette évaluation se fera à l'aune du concept d'ECI après avoir pris soin d'en détailler les aspects principaux.

[1] *Inevitable Collision State*

# 1. Description des méthodes

Le principe général à tous systèmes robotiques mobiles est d'acquérir des données via des capteurs, de les traiter afin de construire un modèle de l'environnement, puis d'agir en fonction de ce modèle. L'acquisition des données n'est pas une caractéristique propre à chaque méthode, c'est donc sur la modélisation et la prise de décision que portera la description des 5 méthodes.

## 1.1. La méthode *Vector Field Histogram**

VFH* fait suite aux améliorations successives de VFH [8] et VFH+ [9] qui sont des méthodes purement réactives. VFH* fait partie des approches hybrides par le fait qu'elle teste les conséquences de plusieurs choix avant de prendre sa décision.

### 1.1.1. VFH*, la modélisation

La modélisation consiste dans un premier temps à représenter l'environnement par une grille d'occupation centrée sur le robot, où chaque cellule contient une valeur entière correspondant à la probabilité d'y trouver un obstacle. Une grande valeur indique que plusieurs mesures indiquent la présence d'un obstacle. Une faible valeur indique au contraire que peu de meure indique une telle présence et qu'il peu ainsi s'agir de mesures parasites. C'est pour cette raison que ces valeurs portent le nom de "valeur de certitude".

Ensuite, la grille d'occupation représentant l'espace autour du robot est convertie en histogramme en le discrétisant en 72 secteurs de 5°. Cette modélisation porte ainsi le nom d'histogramme polaire (figure 2).

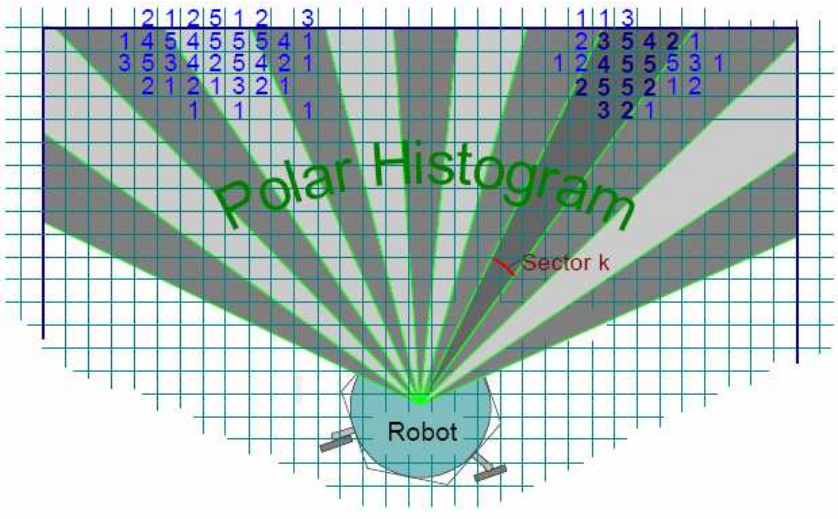


*figure 2 : construction de l'histogramme polaire (polar histogram) via une grille d'occupation*

Pour chaque secteur, les valeurs des cellules qu'il contient et leur distance par rapport au robot sont utilisées pour calculer une valeur de "densité polaire d'obstacle" ou *Polar Obstacle Density* (POD). Les POD sont proportionnelles à la proximité d'un obstacle et à la certitude qu'il existe dans cette direction. La figure 3 représente ces POD sous la forme d'un histogramme (à gauche). Pour aider à la compréhension, les POD de l'histogramme ont été représentées sur la grille d'occupation (figure 3 à droite).

Chaque direction autour du robot est donc représentée par une "barre" de l'histogramme dont la hauteur est proportionnelle à la probabilité d'être proche d'un obstacle.

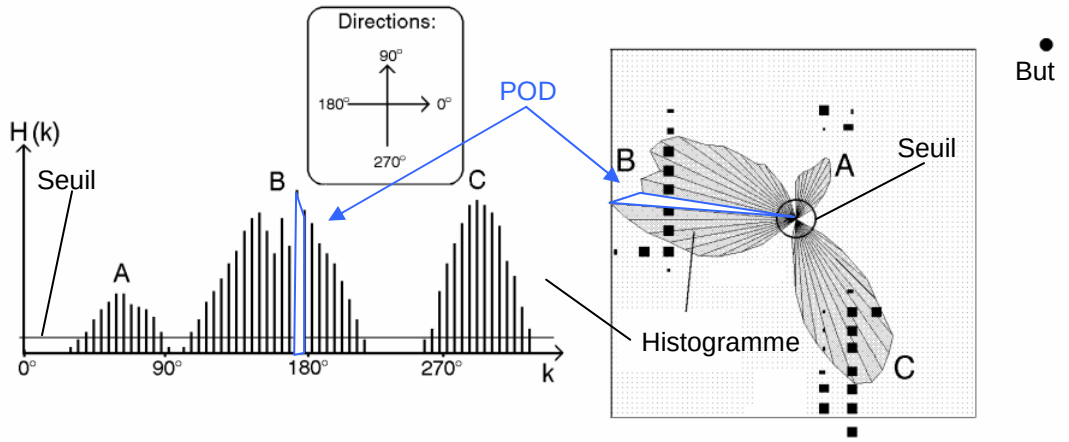


*figure 3 : l'histogramme polaire H permet d'exprimer la densité d'obstacle présent dans un secteur k*

Afin de connaître les zones de passage pouvant être empruntées, les densités polaires sont comparées à un seuil. Le nombre de secteurs consécutifs dont la densité est inférieure au seuil détermine la largeur de ces passages (figure 4).

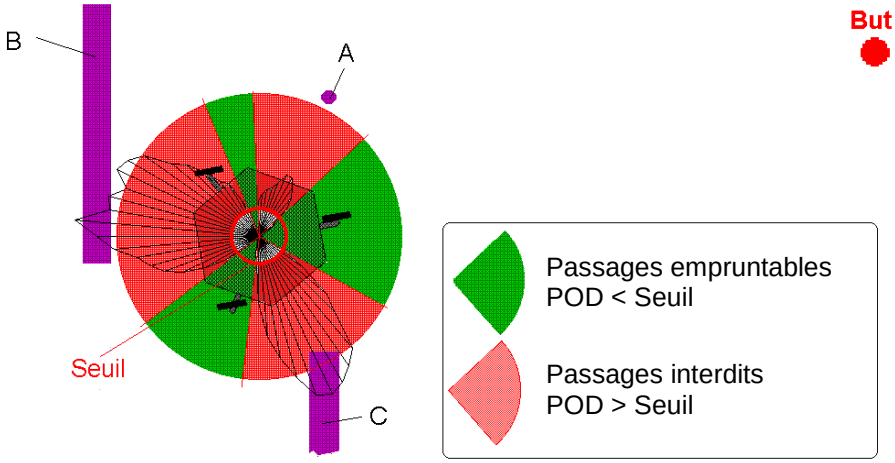


*figure 4 : détermination des zones libres de passage*

### 1.1.2. VFH*, la décision

Une fois l'histogramme polaire construit autour de la position courante du robot, l'algorithme VFH* détermine 1 à 3 directions dans chaque zone de passage libre. Ces directions portent le nom de "directions candidates primaires". Pour chacune de ces directions, VFH* calcule la position et l'orientation qu'aura le robot après s'être déplacé d'un pas de distance $d_s$. L'histogramme polaire est alors recalculé pour chacune de ces positions afin d'en déterminer des "directions candidates projetées". Le procédé se répète de projection en projection afin d'obtenir un arbre de recherche de profondeur notée $n_g$.
Chaque nœud de l'arbre correspond donc à une position et une orientation, et chaque branche représente une distance à parcourir dans une direction donnée. La distance totale projetée $d_t$ est alors déterminée par la relation $d_t = n_g \, . \, d_s$.

Pour prendre en compte (de façon approximative) la dynamique du robot, les branches de l'arbre ne sont pas rectilignes. En effet, plus le changement de direction est fort et plus les branches de l'arbre sont incurvées (figure 5).

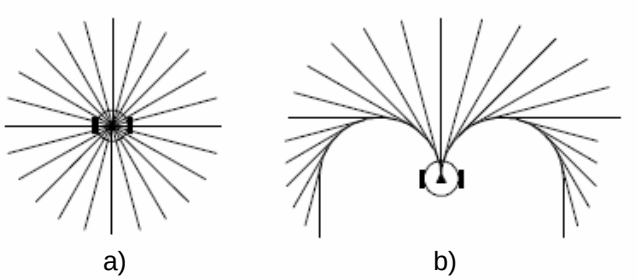


*figure 5 : approximation des trajectoires : a)sans prise en compte de la dynamique du robot, b) avec prise en compte de cette dynamique*

Un coût est ensuite attribué à chaque nœud pour qu'une fonction heuristique puisse choisir la direction candidate primaire la plus appropriée i.e. celle appartenant à l'ensemble des directions menant le plus directement au plus près du but. L'arbre est reconstruit chaque fois que le robot atteint la 1ère position projetée i.e. le 1er nœud de l'arbre.

La figure 6 montre un exemple de chemin parcouru en fonction de la profondeur ($n_g$) de l'arbre. La figure 7 permet de voir les arbres qui ont permis la construction de ces chemins. Les lignes noires indiquent les trajectoires projetées des directions candidates primaires. Les lignes grises montrent les directions dont les coûts des nœuds n'ont pas permis la sélection.

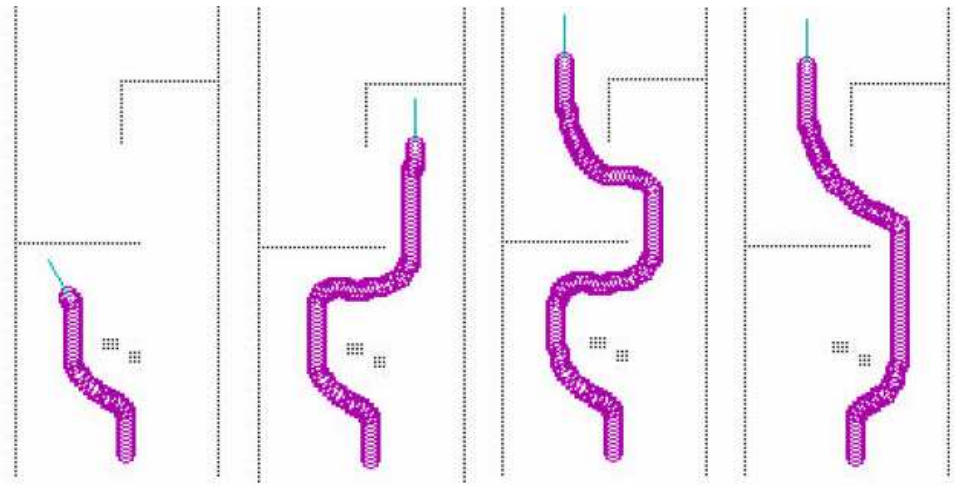


*figure 6 : chemins générés selon la profondeur de l'arbre*

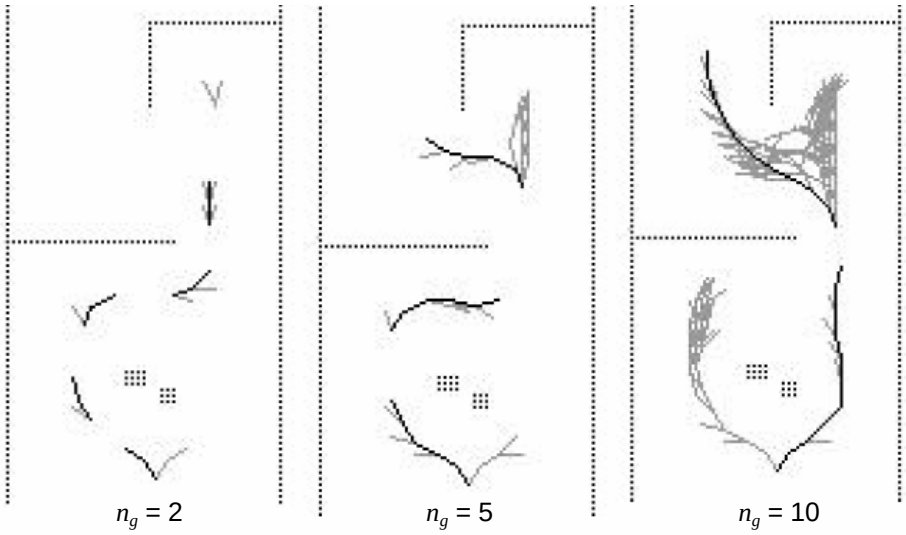


*figure 7 : Arbres générés pour les chemins b), c) et d) de la figure 6*

Durant tout son trajet, VFH* maintient la vitesse de déplacement du robot à son maximum. Cette vitesse est cependant réduite proportionnellement à la proximité d'un obstacle. De même, cette vitesse sera réduite d'autant qu'un changement de direction sera nécessaire.

## 1.2. La méthode *Global Dynamic Window Approach*

GDWA s'appuie sur DWA. En effet, cette dernière étant purement réactive, une fonction permettant de suivre un chemin lui a été ajouté pour pallier aux problèmes de minimums locaux. De ce fait, la modélisation de GDWA est identique à celle de DWA et ces deux méthodes diffèrent lors de la prise de décision.

### 1.2.1. (G)DWA, la modélisation

Le concept général de cette méthode est de calculer une commande plutôt qu'une orientation ou une position. Cette commande, utilisée pour déplacer le robot pendant un intervalle de temps déterminé, est un couple de vitesses angulaire et linéaire. Elle est recalculée pour chaque intervalle de temps. Pour se faire, (G)DWA utilise l'espace des vitesses de translation et de rotation du robot (respectivement $v$ et $\omega$). Au sein de cet espace $Vs$ sont définies des vitesses dites admissibles $Va$ qui permettent au robot de s'arrêter avant d'atteindre un obstacle. L'ensemble de ces vitesses admissibles est défini tel que:

$$V_a = \left\{ (v,\omega) \mid v \leq \sqrt{2 \cdot \mathrm{dist}(v,\omega) \cdot \dot{v}_b} \ \wedge \ \omega \leq \sqrt{2 \cdot \mathrm{dist}(v,\omega) \cdot \dot{\omega}_b} \right\}$$

Où $\dot{v}_b$ et $\dot{\omega}_b$ sont les décélérations de freinage en translation et en rotation. La fonction $dist(v,\omega)$ permet de calculer la distance entre le robot et l'obstacle le plus proche après avoir suivi la trajectoire $(v,\omega)$.

Pour illustrer ce procédé, la figure 8 représente un couloir vu de dessus dans lequel circule le robot et la figure 9 montre comment cet environnement est représenté dans l'espace des vitesses [10].

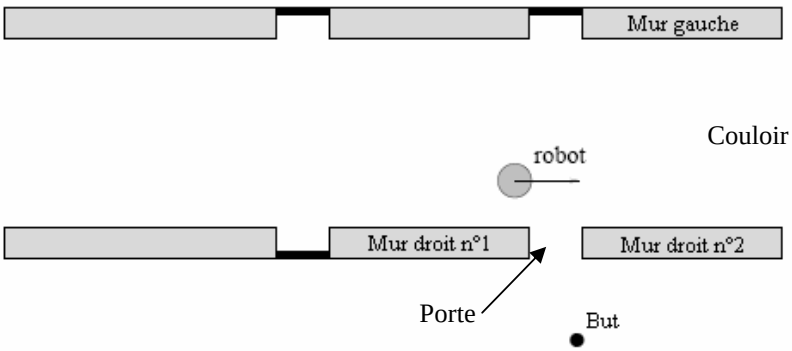


*figure 8 : exemple d'environnement*

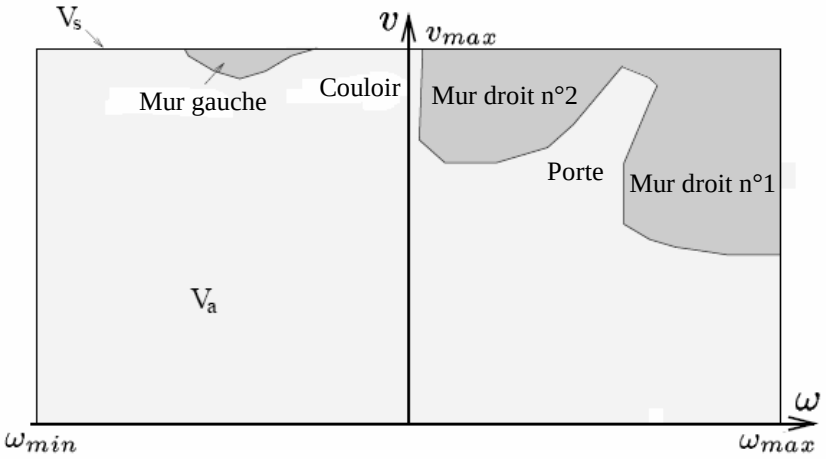


*figure 9 : environnement de la figure 8 traduit dans l'espace des vitesses Vs. Va est l'ensemble des vitesses admissibles*

Compte tenu de la dynamique du robot i.e. sa capacité à accélérer et à décélérer, les vitesses admissibles ne sont pas toutes atteignables dans l'intervalle de temps imparti. Pour exprimer cette contrainte, une fenêtre dynamique est alors générée dans l'espace des vitesses, autour de la vitesse actuelle du robot. Cela réduit ainsi l'espace dans lequel sera recherché la prochaine commande (figure 10).

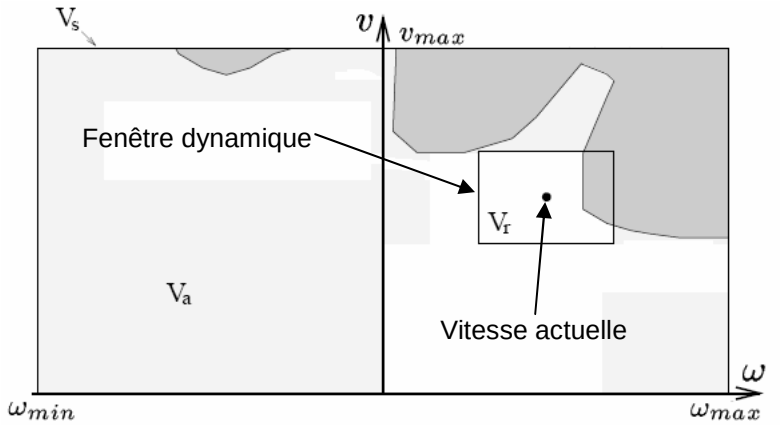


*figure 10 : fenêtre dynamique autour de la vitesse actuelle du robot*

Il reste alors à discrétiser l'espace contenu dans la fenêtre et à y rechercher la prochaine commande. Sans cette discrétisation, l'espace de recherche *Vr* représenterait une infinité de solution.

### 1.2.2. GDWA, la décision

Le processus décisionnel de GDWA est basé sur une fonction NF1 permettant de générer un chemin. NF1 utilise une technique de propagation de vague afin de donner une distance au but pour chaque point de l'espace qu'elle parcourt (figure 11). Une descente de gradient permet alors d'en extraire un chemin.

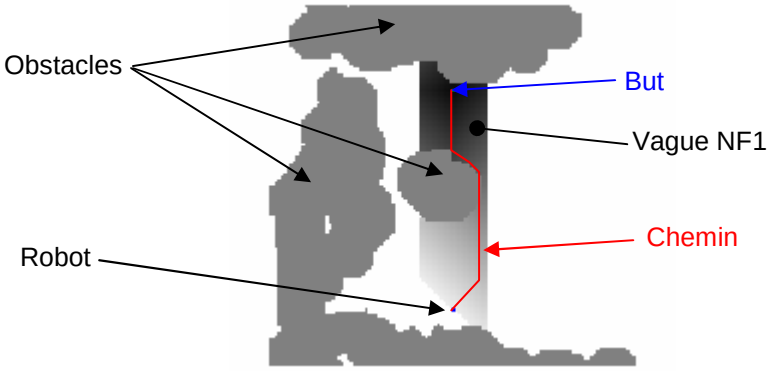


*figure 11 : génération des distances au but par NF1*

Chaque commande de l'espace de recherche $Vr$ est évaluée par une fonction heuristique définie tel que :

$$\Omega_g(\vec{p},\vec{v},\vec{a}) = \alpha \cdot nf1(\vec{p},\vec{v}) + \beta \cdot vel(\vec{v}) + \gamma \cdot goal(\vec{p},\vec{v},\vec{a}) + \delta \cdot \Delta nf1(\vec{p},\vec{v},\vec{a})$$

$\Omega_g$ est la fonction heuristique de GDWA.
$\vec{P}$ est la position courante du robot dans le plan (*x*,*y*).
$\vec{v}$ est la vitesse en translation en *x* et en *y* extraite de $Vr$.
$\vec{a}$ est l'accélération en x et en y extraite de $Vr$.
*nf1()*, *vel()*, *goal()*, *Δnf1()* sont des sous-fonctions telles que :

- *nf1()* valorise les vitesses alignées sur le chemin,
- *vel()* favorise les vitesses élevées lorsque le robot est loin du but et les faibles vitesses lorsqu'il en est proche.
- *goal()* renvoi 1 si la commande à évaluer fait passer le robot sur le but, sinon, elle renvoi 0,
- *Δnf1()* valorise les commandes qui réduisent la distance au but lors du prochain cycle.

$\alpha, \beta, \gamma, \delta$ sont des coefficients permettant de pondérer chaque sous-fonction et ainsi de paramétrer GDWA.
La commande obtenant la meilleure évaluation est choisie pour effectuer le mouvement partiel suivant.

## 1.3. La méthode *Global Nearness Diagram*

A la manière de GDWA, GND étend la méthode ND [11] en utilisant NF1 pour pallier aux problèmes des minimums locaux. Le principe général de (G)ND est d'identifier la situation dans laquelle se trouve le robot parmi 5 prédéfinies et de générer la commande adéquate.

### 1.3.1. GND, la modélisation

Via les capteurs du robot, (G)ND construit 2 diagrammes de proximité[2] sous forme d'histogramme : PND[3] et RND[4]. PND représente la distribution des obstacles autour du robot. RND quant à lui, représente cette distribution par rapport à une zone de sécurité définie autour du robot. La figure 12 illustre l'histogramme PND construit selon un environnement donné. Chaque "barre" représente un secteur et sa hauteur correspond à la proximité de l'obstacle.

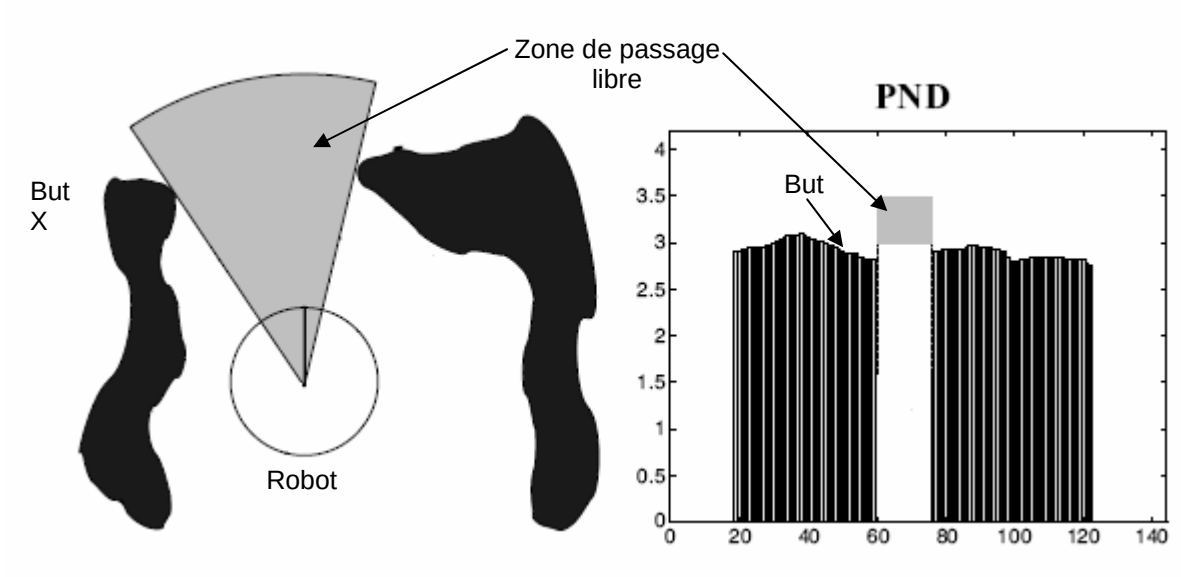


*figure 12 : construction du diagramme PND selon un environnement donné*

[2] Traduit de l'anglais *Nearness Diagram*
[3] PND : *Point Nearness Diagram*.
[4] RND : *Robot Nearness Diagram*.

Dans la figure 13 apparaît la zone de sécurité autour du robot. RND est construit comme PND mais son rôle est différent. Il permet en effet de déterminer si un obstacle pénètre la zone de sécurité.

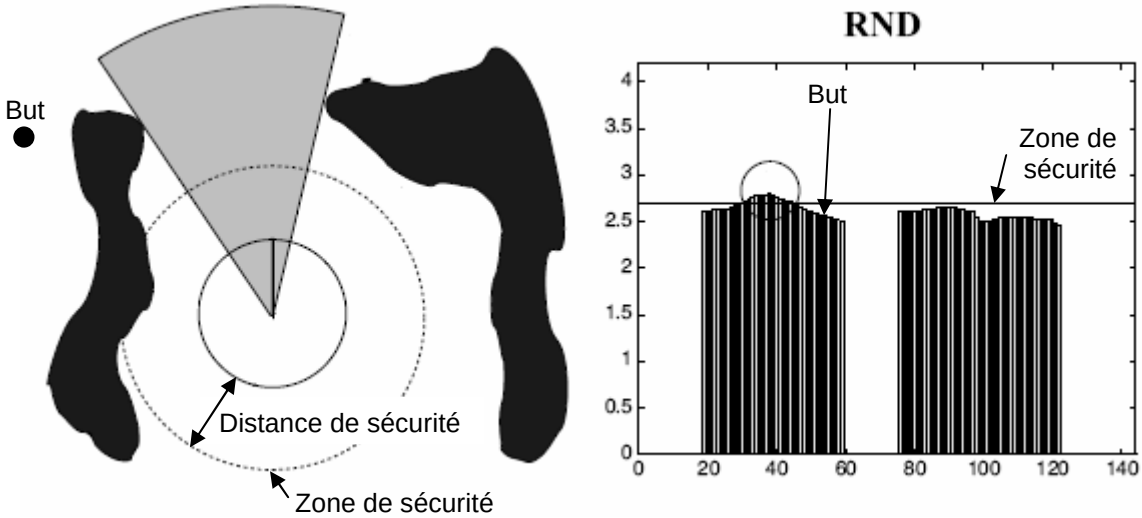


*figure 13 : construction du diagramme RND selon un environnement donné, un obstacle pénètre la zone de sécurité*

Grâce aux constructions de RND et de PND, (G)ND peut alors analyser la situation dans laquelle il se trouve et la reconnaître parmi les 5 prédéfinies. Ces situations sont les suivantes :

1) LS1 (*Low Safety* 1) : au moins un obstacle est présent dans la zone de sécurité mais d'un seul coté de la région libre la plus orientée vers le but (figure 13 & 14 a).
2) LS2 (*Low Safety* 2) : plusieurs obstacles sont présents dans la zone de sécurité mais des 2 cotés de la zone de passage (figure 14 b).
3) HSGR (*High Safety Goal in Region*) : aucun obstacle ne pénètre la zone de sécurité et le but à rejoindre est dans une zone de passage (figure 15 a).
4) HSWR (*High Safety Wide Region*) : aucun obstacle n'est présent dans la zone de sécurité du robot mais le but n'est pas directement atteignable. De plus, la région libre pour se déplacer est large i.e. représente plus d'un quart des secteurs du diagramme (> 90°) (figure 15 b).
5) HSNR (*High Safety Narrow Region*) : idem à la situation HSWR sauf que la région libre représente moins d'un quart des secteurs du diagramme (< 90°) (figure 15 c).

En fonction de la situation dans laquelle se trouve le robot, GND va prendre une décision prédéfinie.

### 1.3.2. GND, la décision

Dans cette méthode, NF1 est utilisée pour créer des "sous-buts" sur lesquels se basera ND pour réagir comme s'il s'agissait du but final.

Chaque situation est associée à une trajectoire afin de produire un comportement adéquat. Ainsi, dans les cas de LS1 et LS2, le but des actions est de mener le robot dans une situation de haute sécurité (HSGR, HSWR ou HSNR) :

1) LS1: éloigner le robot de l'obstacle mais dans la direction la plus orientée vers une région libre proche du but (figure 14 a). La vitesse choisie est alors proportionnellement inverse à la proximité de l'obstacle.
2) LS2: éloigner le robot des obstacles mais dans la direction la plus orientée vers une région libre proche du but (figure 14 b). La vitesse est gérée comme pour LS2.

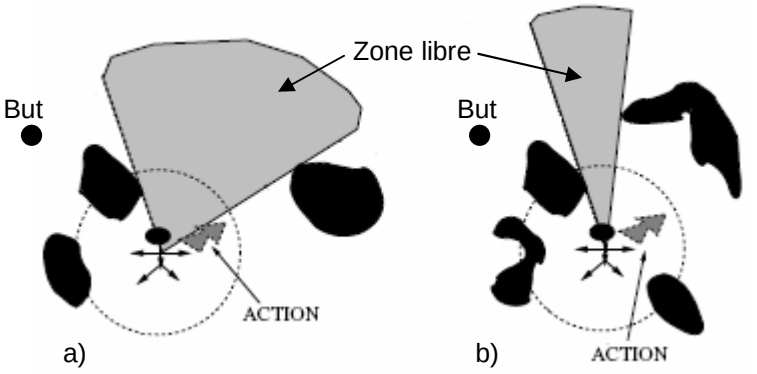


*figure 14 : types d'actions menées dans les situations LS1 et LS2*

3) HSGR: diriger le robot en direction du but à la vitesse maximum (figure 15 a).
4) HSWR: choisir la direction au sein de la zone libre appropriée qui longe l'obstacle (figure 15 b).
5) HSNR: se diriger au centre de la zone libre adéquate (figure 15 c).

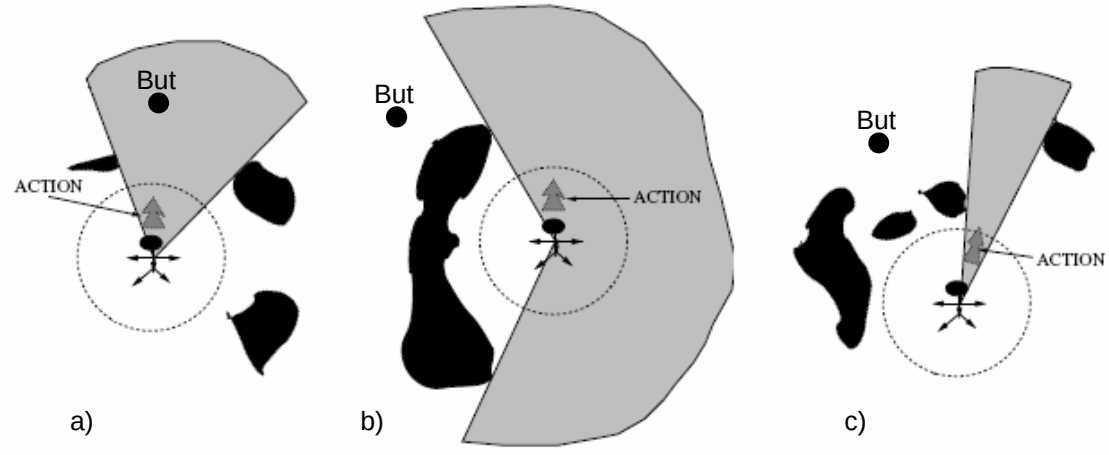


*figure 15 : types d'actions menées dans des situations dites de haute sécurité*

## 1.4. La méthode *Elastic Bande + Dynamic Window Approach*

La méthode hybride traitée maintenant est une combinaison de 3 méthodes distinctes :

- **NF1** : elle permet de générer un chemin menant au but. L'inconvénient de ce chemin est d'être irrégulier et de frôler les obstacles.
- ***Elastic Band*[5] (EB)** : s'appuyant sur le chemin de NF1, elle "lisse" ce dernier et maintient le robot à une certaine distance des obstacles. De plus, elle permet de réagir lorsque des obstacles encombrent le chemin dessiné par NF1.
- **DWA** : elle permet de générer les commandes motrices adéquates.

NF1 n'entrant pas dans le cadre de se rapport, l'analyse qui suit se base donc sur EB+DWA. Il convient de souligner que EB n'est pas non plus une méthode réactive d'évitement mais son étroite interaction avec DWA justifie cette attention.

### 1.4.1. EB+DWA, la modélisation

Les informations issues des capteurs du robot permettent de renseigner le robot sur la position des obstacles qui l'entourent. Cette carte de l'environnement est représentée sous la forme d'une grille d'occupation permettant la construction de la bande élastique. Cette bande se représente par un chapelet de disques se chevauchant appelés bulles. Chaque bulle, d'un diamètre variable, représente un espace libre empruntable par le robot. Elles sont soumises à deux types de forces : répulsives et attractives. Les forces répulsives sont générées par les obstacles et les forces attractives s'exercent uniquement entre les bulles consécutives. Ceci a pour effet de maintenir la bande élastique tendue et éloignée des obstacles.

DWA va ensuite être utilisée pour générer des commandes afin de suivre cette bande élastique. L'utilisation de DWA diffère légèrement par rapport à ce qui a été vu précédemment les obstacles n'y sont pas modélisé et l'espace des vitesses du robot est traduit en espace de vitesses des roue motrices : $\dot{\phi}_l$ et $\dot{\phi}_r$ (figure 16).

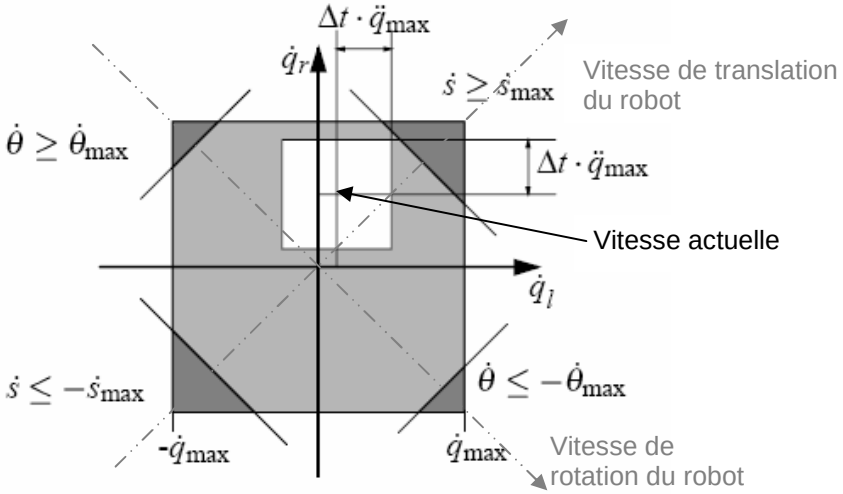


*figure 16 : DWA dans l'espace de vitesse de rotation des roues*

[5] Bande élastique

### 1.4.2. EB+DWA, la décision

Comme dans la section **1.2.2**, DWA choisi la commande adéquate sur l'espace échantillonné grâce à une fonction heuristique. Cette fonction se décompose cependant en 3 sous-fonctions différentes : direction, vitesse, évitement.

- La sous-fonction de direction, valorise la commande lorsqu'elle permet au robot de suivre la bande élastique,
- celle de la vitesse permet la sélection de la plus grande vitesse,
- et enfin, la sous-fonction d'évitement favorise les commandes qui maximisent la distance entre le robot et les obstacles. En effet, via les données de la bande élastique, elle calcule si le temps avant collision induit par cette future commande permettra au robot de pouvoir freiner. Si ce temps avant collision est inférieur au temps de freinage nécessaire, la commande est déclarée inadmissible.

La somme pondérée de ces sous-fonctions permet alors de choisir la prochaine commande.

Lorsqu'une bulle de la bande élastique est contrainte de rétrécir en dessous d'un diamètre minimum, elle disparaît et la bande élastique devient alors invalide. Cela permet d'indiquer à au système qu'un chemin doit être recalculé et une nouvelle bande est alors créée. Pendant ce calcul, DWA continue de suivre l'ancienne bande élastique.
Pour éviter que la bande ne soit pas coupée inutilement par des obstacles lointains, ceux-ci sont volontairement masqués lorsqu'ils se trouvent au-delà d'une certaine distance du robot.

## 1.5. La méthode *Velocity Obstacle* du système MAid

MAid est un robot de type chaise roulante combinant 2 systèmes de navigation : NAN[6] et WAN[7]. NAN permet d'aider l'usager à évoluer dans un environnement étroit et encombré. C'est un système semi autonome qui sort donc du cadre de cette étude. WAN quant à lui est une navigation autonome mettant en œuvre le concept des *Velocity Obstacle*.

### 1.5.1. VO, la modélisation

Dans un 1er temps, cette méthode distingue les objets les uns par rapport aux autres. Ensuite, elle détermine ceux qui sont statiques et ceux qui sont mobiles. Ce processus est mis en oeuvre en comparant la perception actuelle et celle qui précède.
Avec la connaissance des différentes positions occupées par un objet à un instant *t* et à un instant *t-1*, le système en déduit une direction et une vitesse i.e. une trajectoire (figure 17).

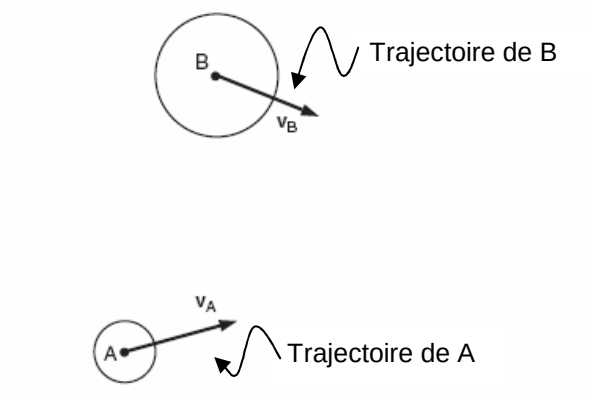


*figure 17 : A le robot et B l'obstacle*

WAN calcule ensuite un cône de collision $CC_{AB}$ correspondant aux trajectoires de $A$ relatives à $B$ menant à une collision. La figure 18 illustre le robot et un obstacle B dans l'espace des vitesses. Il est à noter que la prise en compte de la géométrie du robot est faite en le réduisant en un point et en grossissant les obstacles du rayon du robot.
Tout point de cet espace représente une vitesse. $V_{a\ col}$ est un exemple de vitesse conduisant le robot à entrer en collision avec B. Il est à noter que plus cette vitesse est grande, et plus tôt aura lieu la collision. A l'inverse, réduire cette vitesse augmentera proportionnellement le temps avant collision. Toujours dans le cas de la figure 18, $V_{ar}$ est un exemple de vitesse permettant d'éviter la collision avec B en passant "derrière" celui-ci, alors que $V_{af}$ permet cet évitement en passant "devant".

[6] *Narrow Area Navigation*
[7] *Wide Area Navigation*

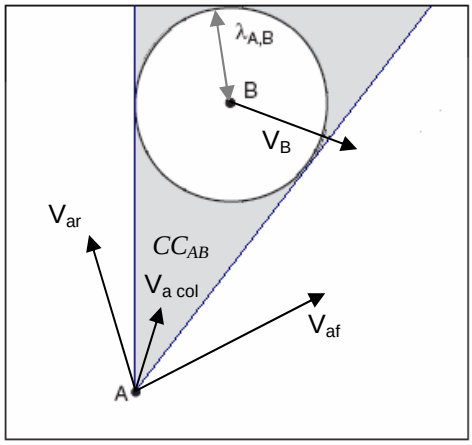


*figure 18 : cône de collisions entre A et B*

La construction des vitesses obstacles (VO) se fait ensuite en translatant le cône de collision d'un obstacle par le vecteur de sa vitesse. L'intérieur de $VO_B$ (figure 19) représente ainsi les vitesses futures menant à une collision avec B.

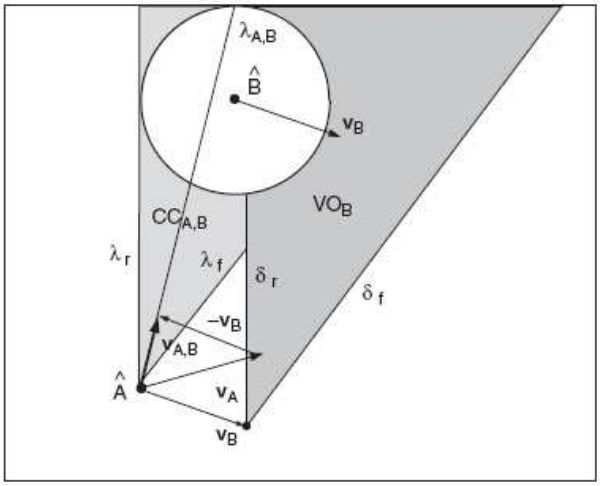


*figure 19 : translation du cône de collisions*

Le VO total d'un environnement encombré de plusieurs obstacles sera l'union des VO de chacun d'eux (figure 20). Les vitesses évitant les collisions sont celles situées hors des VO (zone blanche de la figure 20).

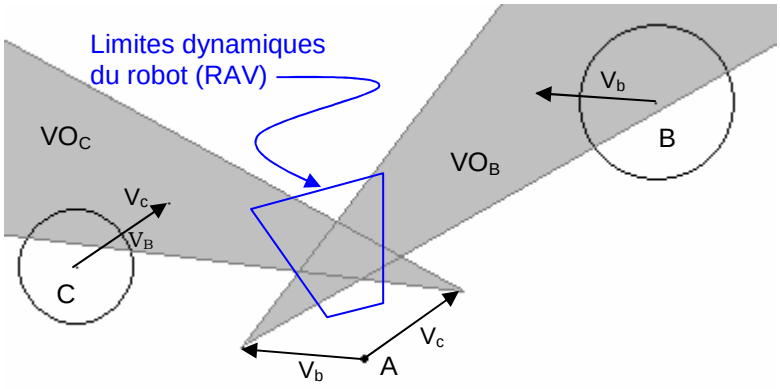


*figure 20 : VO de 2 obstacles mobiles*

Toutefois, cet ensemble de vitesses d'évitement doit être limité vis-à-vis des contraintes dynamiques du robot. Un sous-ensemble de vitesses valides (RAV[8]) est donc calculé géométriquement en superposant aux VO un polygone représentatif de ces contraintes dynamiques (figure 20, polygone bleu). Ainsi, seules les vitesses hors des VO et à l'intérieur du polygone peuvent êtres mises en œuvre par le robot dans le temps imparti.

### 1.5.2. VO, la décision

Les VO sont continuellement calculés par le système et la trajectoire à suivre est sélectionnée au terme de chaque calcul. Comme pour GDWA, c'est une fonction heuristique qui choisit la trajectoire adéquate en satisfaisant les critères suivants :

- être proche du centre de l'ensemble RAV,
- être la plus dirigée vers le but,
- préférer les trajectoires qui ne coupent pas celles des obstacles.

De plus, les trajectoires évitant les collisions imminentes sont privilégiées. En effet, les VO sont basés sur une approximation linéaire des trajectoires[9], et sont donc sujettes à être erronée au fil du temps. Pour cette raison, un horizon de temps $T_h$ paramétrable a été introduit. Seules les collisions pouvant avoir lieu dans une période de temps $t < T_h$ sont prises en compte (figure 21). Cela revient graphiquement à tronquer les VO correspondants ($VO_C$).

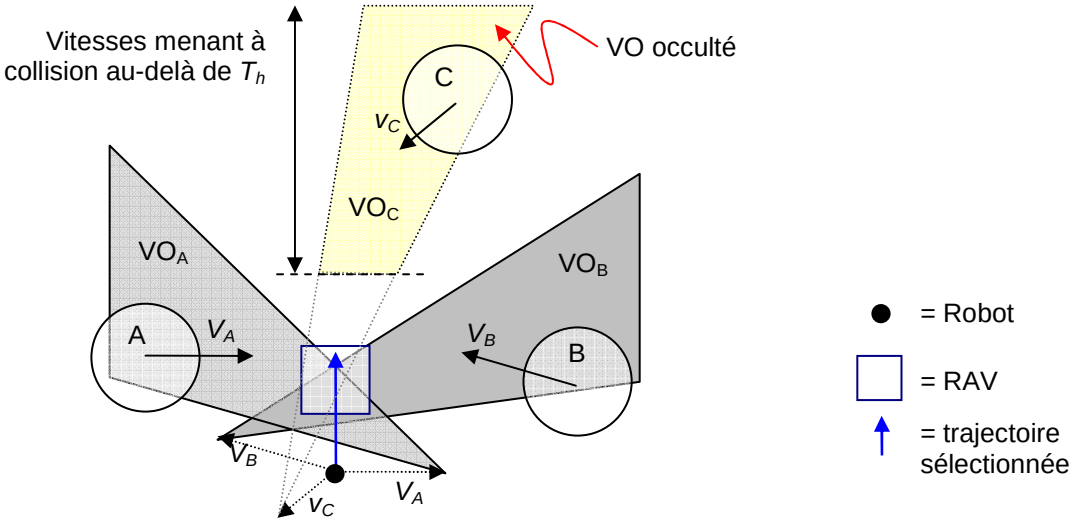


*figure 21 : les VO entraînant des collisions au-delà d'un temps $T_h$ sont occultés*

[8] *Reachable Velocity Obstacle* (RAV)
[9] Une trajectoire est dite linéaire quand sa direction et sa vitesse ne varient pas dans le temps

# 2. Evaluation de la sûreté des méthodes à l'aune d'ECI

Sachant comment fonctionnent ces 5 méthodes, il est maintenant possible d'en évaluer leur sûreté au sein d'un environnement dynamique. Pour ce faire, il convient d'expliquer comment utiliser le concept d'état de collision inévitable.

## 2.1. Le concept d'état de collision inévitable

Le concept de collision inévitable définit tout état pour lequel, une collision sera inévitable quoi que soit les décisions futures. Il se décompose en trois règles :

***1)*** prendre en compte la dynamique du robot,
***2)*** prendre en compte la trajectoire des obstacles,
***3)*** ne pas restreindre sa décision à un horizon temporel borné.

***1)*** De par ses contraintes cinématiques, son inertie, et sa vitesse, un robot en mouvement a besoin d'une distance et d'un temps pour pouvoir s'arrêter. Cette notion peut s'illustrer par l'exemple suivant (figure 22). Soit $A$, une particule se déplaçant vers la droite uniquement et à vitesse variable $v$. La dynamique de cette particule est telle qu'il lui faut une distance $d(v)$ pour s'arrêter. Si $A$ se retrouve à une distance inférieure à $d(v)$, une collision aura lieu inévitablement. $A$ est alors dans un état de collision inévitable.

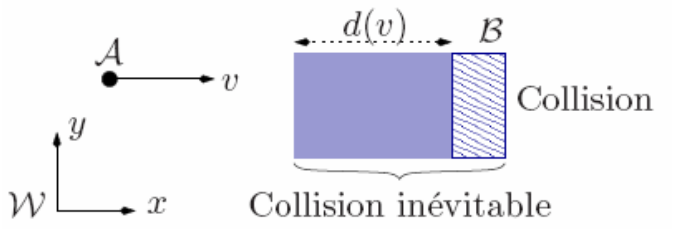


*figure 22 : représentation du concept de collision inévitable dans l'espace de travail W*

***2)*** La trajectoire d'un obstacle mobile est à considérer pour l'éviter. En effet, si dans notre exemple, la particule $A$ est capable de changer de direction, la trajectoire 1 (figure 23) mène à une collision alors que la trajectoire 2 permet l'évitement.

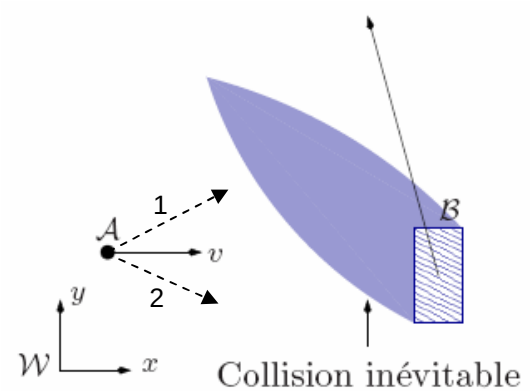


*figure 23 : un obstacle hostile rend le robot en danger même s'il est passif*

***3)*** Restreindre sa décision à un horizon temporel borné, c'est exclure l'ensemble des situations pouvant se produire au-delà de cet horizon. En résumé, avoir un raisonnement à court terme peut avoir des conséquences néfastes à long ou moyen terme. La
figure 24 [7] illustre ce principe. Si le système se projette à l'horizon de temps *h1*, il en déduit que D est une trajectoire aboutissant à une collision. Il est alors amené à choisir A, B ou C. Cependant, B et C sont des trajectoires menant à une collision au-delà de *h1*, il se sera donc mis en danger. Cette illustration montre aussi qu'augmenter l'horizon temporel n'est pas suffisant car si le système calcule ce qui va se passer jusqu'à *h2*, il aura encore le choix entre A et B. Et seul A mène au but de façon sûre. La solution pour garantir la sûreté est alors d'utiliser un horizon temporel infini ou au moins supérieur au temps requis pour atteindre le but (en considérant qu'il soit sûr).

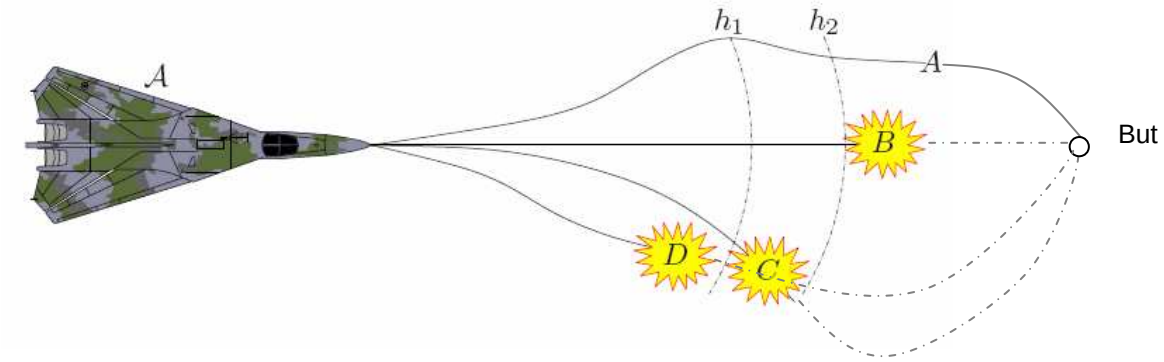


*figure 24 : les dangers d'un raisonnement borné dans le temps*

## 2.2. Evaluation des méthodes

Du moment où l'une des trois règles n'est pas respectée, le système considéré peut tôt ou tard se retrouver dans une situation de collision inévitable. L'évaluation des 5 méthodes de navigation autonome va donc consister à valider leur respect de ces trois règles.

### 2.2.1. Evaluation de *Vector Field Histogram**

VFH* modélise de la même manière tous les obstacles perçus. S'ils sont dynamiques, la modélisation de cette méthode ne peut alors en aucun cas tenir compte de leur trajectoire. De plus, la fonction heuristique privilégiant les faibles changements de direction ainsi que les trajectoires menant au but, VFH* peut tout à fait placer le robot sur la trajectoire d'un mobile, risquant ainsi la collision (figure 25). En outre, la gestion de la vitesse par VFH* est telle qu'elle diminue lorsque le robot s'approche d'un obstacle. En d'autre terme, si un obstacle lui est hostile[10], le robot tend à s'immobiliser.

[10] Sa trajectoire converge vers le robot

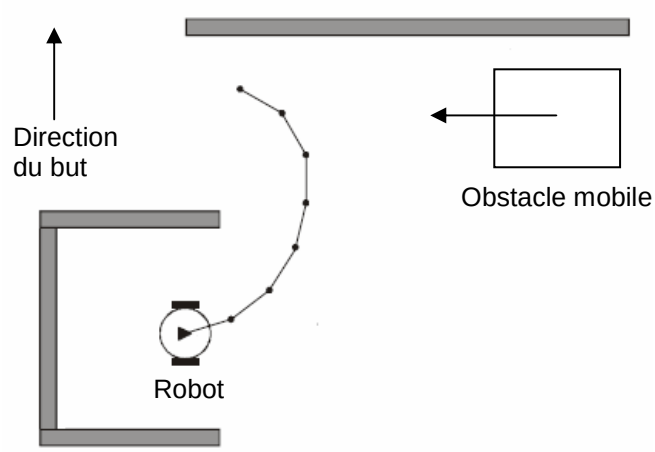


*figure 25 : VFH* peut mettre le robot sur la trajectoire d'un mobile*

De plus, les choix de VFH* s'appliquent pour un déplacement d'une distance déterminée ($d_s$). Ce qui revient à raisonner sur un horizon temporel borné.

### 2.2.2. Evaluation de *Global Dynamic Window Approach*

Tout comme VFH*, GDWA ne différencie pas la nature des obstacles dans sa modélisation. Les mobiles sont donc vus comme statiques. Dès lors, l'ensemble des vitesses admissibles souffre d'une erreur proportionnelle à la vitesse de l'obstacle. De plus, la fonction d'objectif (qui sélectionne les commandes) favorise le suivi de NF1 et les hautes vitesses. Or NF1 est une fonction délibérative qui a le désavantage de frôler les obstacles.

L'absence de prise en compte de la dynamique des obstacles et la propension à les frôler ne peut donc pas assurer la sûreté de cette méthode. De plus, GDWA fait des choix pour chaque cycle de calcul sans en connaître les conséquences d'un point de vue dynamique. Il peut ainsi se mettre sur la trajectoire d'un obstacle mobile.

### 2.2.3. Evaluation de *Global Nearness Diagram*

Tout le processus de modélisation de cette méthode ne fait pas la distinction entre les objets mobiles et statiques ignorant ainsi tout de leur trajectoire. De plus, comme pour VFH*, GND réduit sa vitesse à mesure qu'il s'approche d'un obstacle, se mettant ainsi en danger si l'obstacle est hostile. Par exemple, dans le cas d'une situation de type LS1, GND va bifurquer pour s'éloigner de l'obstacle et rien ne l'empêche de suivre une direction dangereuse (figure 26).

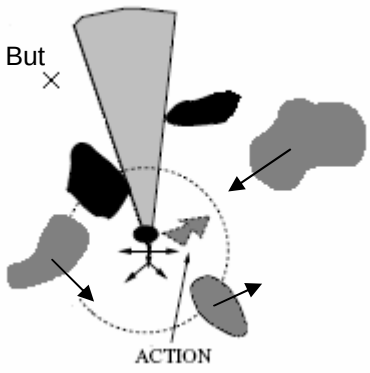


*figure 26 : GND ne sait pas prévoir de collision*

### 2.2.4. Evaluation de la bande élastique combinée avec DWA

La bande élastique permet de maintenir le chemin éloigné des obstacles autant que possible. Cependant elle ne tient pas compte de la dynamique des obstacles, et occulte même certain d'entre eux pour éviter d'invalider la bande. En outre, une bande invalide est toujours suivie par DWA en attendant que la nouvelle bande soit générée. Cette méthode ne garantit donc pas la sûreté.

### 2.2.5. Evaluation de la méthode des *Velocity Obstacle*

Le concept des VO est le seul qui modélise la dynamicité des obstacles. La méthode étudiée qui l'implémente (WAN) est en effet capable de choisir des trajectoires en fonction de la vitesse du robot, de ses contraintes dynamiques, et en fonction de la trajectoire des obstacles. En d'autres termes, cette méthode de navigation satisfait les 2 premières règles du concept d'ECI énoncées plus haut. Cependant, WAN choisi une trajectoire selon un horizon temporel borné. En effet, elle occulte volontairement les vitesses menant à une collision au-delà d'un temps prédéfini. Dès lors, la trajectoire sans collision générée à un instant $t$ peut mener le système dans une situation à $t+1$ où l'ensemble des vitesses de RAV ne contient pas de solution (figure 27). Cette méthode ne garantit donc pas la sûreté.

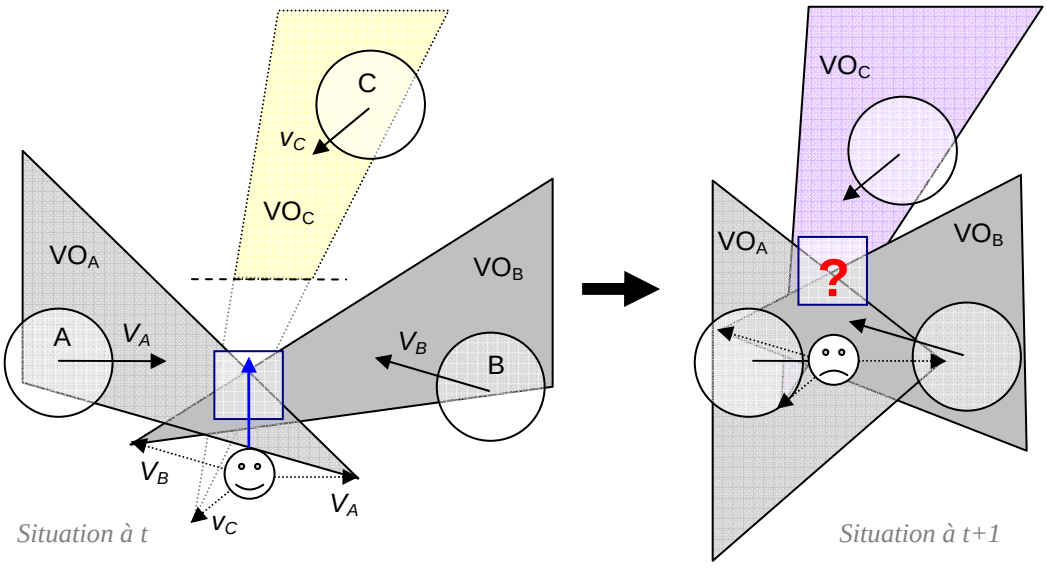


*figure 27 : VO implémenté avec un raisonnement à court terme*

## Conclusion

Nous avons pu apercevoir la diversité actuellement existante dans le domaine des méthodes de navigation autonome. Ces concepts ont différentes façons d'aborder la modélisation de l'environnement et d'y réagir. Au cours de ce rapport, nous avons vu par exemple des approches qui modélisent l'environnement sous formes d'objets dynamiques (VO) et d'autres qui traduisent l'environnement perçu dans un espace de vitesse (DWA).

Toutes ces méthodes analysées présentent des facettes intéressantes de par leur conception. Mais elles toute un point commun : leur incapacité à assurer la sûreté dans un environnement dynamique. Ainsi, si les expérimentations et mises en application de ces méthodes donnent des résultats satisfaisants il est intéressant d'en resituer le contexte :

- un environnement statique pour VFH*, GDWA et GND,
- une vélocité du robot permettant de négliger les distances d'arrêt ou d'évitement (VFH* et GND notamment),
- un environnement où les obstacles vont consciemment et volontairement éviter les collisions avec le robot pour VO et EB+DWA.

On constate alors qu'une série de tests plus ou moins importante ne peut pas permettre de garantir de la sûreté d'une méthode

Cette évaluation menée grâce à l'utilisation du concept d'état de collision inévitable illustre qu'il définit parfaitement la notion de sûreté dans sa généralité. En effet la sûreté n'est plus simplement l'état de non collision du robot avec son environnement, mais bien, la capacité à ne pas se mettre dans un état qui le conduira inévitablement à une collision.

Ce concept n'a jamais été usité lors de la conception des méthodes étudiées dans ce rapport. C'est véritablement avec ce concept d'ECI que les méthodes pourront donner des résultats concluants vis-à-vis de la sûreté en environnement dynamique. C'est d'ailleurs la raison pour laquelle des recherches actuelles se basent sur ce concept. Sera-t-il utilisé comme critère commun aux nouvelles méthodes de navigation autonomes?

## Bibliographie

[1] LARGE Frédéric. *Navigation autonome d'un robot mobile en environnement dynamique et incertain*. Thèse en informatique et robotique avancée. Université de Savoie. Soutenue à Grenoble le 5 novembre 2003. 121p.

[2] BORENSTEIN Johann, ULRICH Iwan. *VFH* : Local Obstacle Avoidance with Look-Ahead Verification*. Séance de la conférence internationale IEEE sur la robotique et l'automatisation. San Francisco, 24 au 28 avril 2000. p. 2505-2511.

[3] BROCK Oliver, KHATIB Oussama. *High-Speed Navigation Using the Global Dynamic Window Approach*. Séance de la conférence internationale IEEE sur la robotique et l'automatisation. Detroit, mai 1999. p. 341-346.

[4] MINGUEZ J., MONTANO L., SIMEON T. *et al*. *GLOBAL NEARNESS DIAGRAM NAVIGATION (GND)*. Séance de la conférence internationale IEEE sur la robotique et l'automatisation. Séoul, 21 au 26 mai 2001. p. 33-39.

[5] PHILIPPSEN Roland, SIEGWART Roland. *Smooth and efficient Obstacle Avoidance for a Tour Guide Robot*. Séance de la conférence internationale IEEE sur la robotique et l'automatisation. Taipei, 2003. 6 p.

[6] PRASSLER Erwin, SCHOLZ Jens, FIORINI Paolo. *A Robotic Wheelchair for Crowded Public Environnement*. IEEE Robotics & Automation Magazine. Mars 2001, Vol. 8, p. 38-45

[7] FRAICHARD Thierry. *Contribution à la planification de mouvement*. Mémoire de diplôme d'habilitation à diriger des recherches, Inst. Nat. Polytechnique de Grenoble, Grenoble, 13 janvier 2006. 69 p.

[8] BORENSTEIN Johann, KOREN Yoram. *The vector field histogram – fast obstacle avoidance for mobile robots*. IEEE Journal of Robotics and Automation. Juin 1991, Vol. 7, n°3, p. 278-288.

[9] BORENSTEIN Johann, ULRICH Iwan. *Reliable obstacle avoidance for fast mobile robots*. Séance de la conférence internationale IEEE sur la robotique et l'automatisation. Leuven, 16 au 21 mai 1998. p. 1572-1577.

[10] FOX Dieter, BURGARD Wolfram, THRUN Sebastian. *The Dynamic Window Approach to Collision Avoidance*. IEEE Robotics & Automation Magazine. Mars 1997, Vol. 4, 23p.

[11] MINGUEZ Javier, MONTANO Luis. *GLOBAL NEARNESS DIAGRAM NAVIGATION (GND)*. IEEE Transactions on robotics and automation. Février 2004, Vol. 20 n°1.

[12] PRASSLER Erwin, SCHOLZ Jens, FIORINI Paolo. *MAid : a robotic wheelchair roaming in e railway station*. Disponible sur : http://voronoi.sbp.ri.cmu.edu/~fsr/FINAL_PAPERS/27_Prassler.pdf (consulté le 2 mars 2006).

**C.N.A.M : EPREUVE PROBATOIRE en INFORMATIQUE**

Evaluation de la sûreté de techniques de navigation autonomes

LAFORET Stéphane Grenoble, le 14 mars

---

Résumé :

Pour permettre le déplacement autonome d'un robot dans un environnement dynamique et incertain, des méthodes combinant des concepts délibératifs et réactifs sont apparues : les méthodes hybrides. 5 d'entres elles les plus usitées sont décrites dans ce rapport afin de pouvoir en évaluer la sûreté à l'aune du concept d'état de collision inévitable. Ce concept définit la sûreté dans sa généralité et se révèle être un instrument efficace pour évaluer objectivement leur garantie de non collision. Ainsi, il apparaît que les méthodes étudiées dans ce document ne garantissent pas la sûreté dans un environnent dynamique.

---

Mots-clés : robot, sûreté, état de collision inévitable, méthodes hybrides, VFH*, GND, GDWA, RoboX, MAid.

---

Keywords : robot, safety, inevitable collision state, hybrid method, VFH*, GND, GDWA, RoboX, MAid.

---