

# Comment asservir une base roulante dans un espace avec obstacles?

Justin Cano

École Centrale de Marseille  
Club Robotique E-Gab

*justin.cano@centaliens-med.fr*

V3.1 – 14 mars 2025.

Partie II - Contrôle de la base roulante et calcul de trajectoire

- 1 Stratégie d'explication
- 2 Cinématique du robot
- 3 Contrôleur point à point
- 4 Calcul de trajectoires

# Contenu de la section

- 1 Stratégie d'explication
- 2 Cinématique du robot
- 3 Contrôleur point à point
- 4 Calcul de trajectoires

Pour aller d'un point A à un point B, que faut-il faire ?

- ➊ Savoir où on se trouve (navigation, autre formation) ;
- ➋ Calculer une trajectoire  $A_1, A_2 \dots A_{N-1}, B$  en prenant compte de son environnement ;
- ➌ Parcourir les segments  $A_i, A_{i+1}$  efficacement ;
- ➍ Savoir comment on se déplace, *ie la cinématique du robot* ;
- ➎ Avoir une vitesse de déplacement asservie (déjà fait, les roues le sont !).

# Contenu de la section

- 1 Stratégie d'explication
- 2 Cinématique du robot
  - Présentation du robot différentiel
  - Hypothèse non-holonyme
  - Cinématique d'un montage différentiel
- 3 Contrôleur point à point
- 4 Calcul de trajectoires

# Présentation du robot différentiel

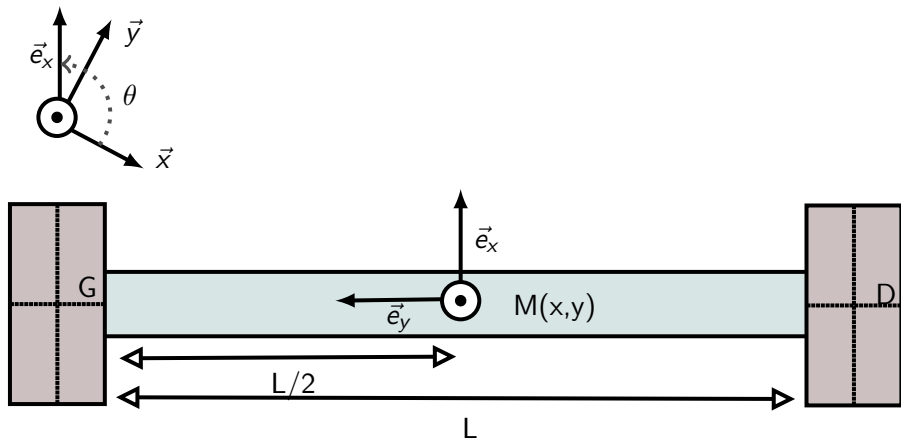


Figure – Schéma du robot, vue de dessus

## États d'intérêt du robot

Comment placer un robot dans un plan ?

- Position  $x, y$  ;
- Angle (ou attitude)  $\theta$ .

# Contrainte non-holonyme

## États d'intérêt du robot

Comment placer un robot dans un plan ?

- Position  $x, y$  ;
- Angle (ou attitude)  $\theta$ .

## Hypothèse de contrainte non-holonyme

Le robot roule sans glisser :

$$\vec{v} \cdot \vec{e}_y = 0$$



# Contrainte non-holonyme

## États d'intérêt du robot

Comment placer un robot dans un plan ?

- Position  $x, y$  ;
- Angle (ou attitude)  $\theta$ .

## Hypothèse de contrainte non-holonyme

Le robot roule sans glisser :

$$\vec{v} \cdot \vec{e}_y = 0$$

## Conséquence

On perd un degré de liberté pour la cinématique.

Seule la vitesse de translation suivant  $\vec{e}_x$  est pertinente. On parlera de **vitesse tangentielle**  $V$ .

## Modèle cinématique holonome

On projette la vitesse  $V\vec{e}_x$  dans le repère général, et on définit la vitesse de rotation  $\Omega$  :

$$\begin{cases} \dot{x} = V \cos \theta \\ \dot{y} = V \sin \theta \\ \dot{\theta} = \Omega \end{cases}$$

# Cinématique d'un montage différentiel II

Question : que valent les vitesses au niveau des roues ?

$$\vec{V}_g = \vec{V}_M + \vec{\Omega} \times \vec{GM} = V\vec{e}_x + \Omega\vec{e}_z \times \left(-\frac{L}{2}\vec{e}_y\right) = \left(V + \frac{\Omega L}{2}\right)\vec{e}_x = \omega_g R_g \vec{e}_x$$

$$\vec{V}_d = \vec{V}_M + \vec{\Omega} \times \vec{DM} = V\vec{e}_x + \Omega\vec{e}_z \times \left(+\frac{L}{2}\vec{e}_y\right) = \left(V - \frac{\Omega L}{2}\right)\vec{e}_x = \omega_d R_d \vec{e}_x$$

Ainsi on peut inverser relation et trouver la vitesse tangentielle en fonction des vitesses de rotation des roues :

$$V = \frac{\omega_g R_g + \omega_d R_d}{2},$$

et pour la vitesse de rotation, nous avons :

$$\Omega = \frac{\omega_g R_g - \omega_d R_d}{L}.$$

**On peut relier vitesse des roues et états cinématique : étape 4 complétée !**

# Contenu de la section

1 Stratégie d'explication

2 Cinématique du robot

3 Contrôleur point à point

- Présentation du contrôleur point à point
- Propriétés du contrôleur
- Correcteur pour le cap
- Quand atteint on le point  $A_i$  et l'angle  $\theta_d$  ?

4 Calcul de trajectoires

# Présentation du contrôleur point à point I

L'idée est ici de nous ramener dans un cas polaire<sup>1</sup>. On peut définir la distance euclidienne au point désiré  $\mathbf{p}_d = [x_d, y_d]^T$  :

$$d = \sqrt{(x_d - x)^2 + (y_d - y)^2}$$

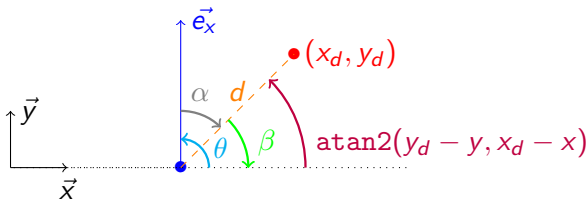
L'angle distant entre «le cap» du robot et le point désiré :

$$\alpha = \text{atan2}(y_d - y, x_d - x) - \theta$$

L'angle entre «le cap» et le vecteur  $\vec{x}$  :

$$\beta = -\theta - \alpha = -\text{atan2}(y_d - y, x_d - x)$$

# Présentation du contrôleur point à point II



On pourra donc boucler (on enverra ces commandes aux roues avec les relations cinématiques trouvées précédemment) le système avec trois gains  $k_d, k_\alpha, k_\beta$  :

$$\begin{cases} V = k_d d \\ \Omega = k_\alpha \alpha + k_\beta \beta \end{cases}$$

1. Ce contrôleur provient d'un article d'*Astolfi et al.* de 1999, voir le mémo compagnon.

- Ce contrôleur est non linéaire ;
- Une analyse de stabilité locale (linéarisation) donne les conditions nécessaires de suivantes pour synthétiser un contrôleur stable :

$$\begin{cases} k_d > 0, \\ k_\beta < 0, \\ k_\alpha > k_\beta; \end{cases}$$

- Ce dernier ne fait pas (encore) le contrôle du cap, on va rajouter un petit quelque chose...

Une fois l'objectif atteint, on fixe le cap avec la simple loi linéaire :

$$\begin{cases} V = 0 \\ \Omega = k_p(\theta - \theta_d) \end{cases}$$

ce contrôleur est stable pour  $k_p > 0$  et non excessif (qui ne fait pas saturer les moteurs). Rappelons que l'erreur en régime permanent est une fonction décroissante de  $k_p$  : un compromis doit s'opérer.



## Arrêt du correcteur non linéaire

Si  $(x - x_d)^2 + (y - y_d)^2 < \rho^2$  avec  $\rho$  un seuil de distance donné, alors on enclenche la correction de cap.

## Arrêt du correcteur de cap

Si de plus  $(\theta - \theta_D)^2 < \gamma^2$  avec  $\alpha$  un seuil d'angle donné, alors on considère l'objectif atteint  $\Rightarrow V = 0$  et  $\Omega = 0$ .

**L'objectif 3 est complété : on est capable de parcourir un segment.**

Regardons-le et exécutons un exemple.

Un schéma-blocs `Simulink`, dans le répertoire `controleur_pt_a_pt` montre la logique d'asservissement à implémenter.

# Contenu de la section

- 1 Stratégie d'explication
- 2 Cinématique du robot
- 3 Contrôleur point à point
- 4 Calcul de trajectoires
  - Stratégie
  - Dilatation de la carte d'obstacles
  - Planification de trajectoire par descente de gradient
  - Lissage des courbes

On cherche à trouver une suite de points dans l'espace d'état :

$$X_d^1, X_d^2, \dots, X_d^{N-1}$$

avec  $X_d = [x_d, y_d, \theta_d]^\top$ , on peut laisser libre l'orientation toutefois. Notre premier souci sera d'éviter les obstacles.

## Stratégie

- 1 Recensement des obstacles ;
- 2 Dilatation de ces derniers d'une zone de sécurité autour du centre de masse ;
- 3 Méthode des potentiels ;
- 4 Lissage des courbes.

# Dilatation de la carte d'obstacles I

Soit un espace de travail connu :

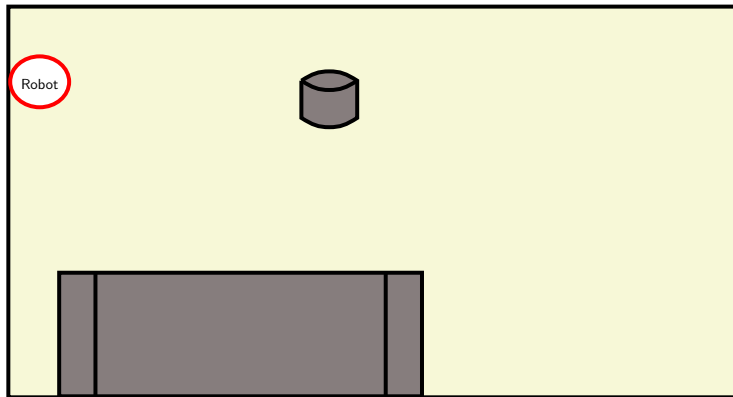


Figure – Environnement connu.

# Dilatation de la carte d'obstacles II

En supposant que le robot soit inscrit dans un cercle de rayon  $R$ , nous dilatons les obstacles de ce rayon.

⇒ le robot en devient donc un point aux yeux des "obstacles".

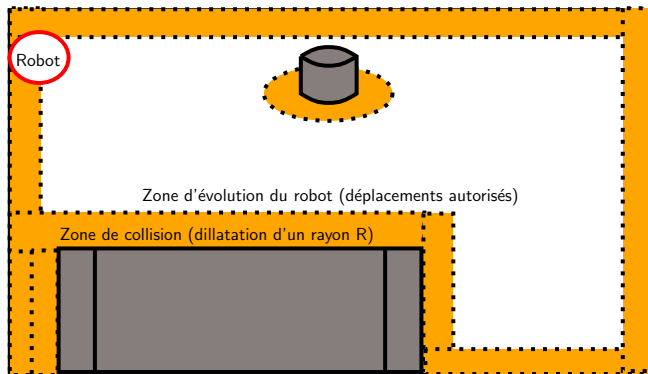


Figure – Environnement dilaté.

# Dilatation de la carte d'obstacles III

On peut donc en extraire la carte binaire d'occupation des obstacles : cette carte doit être vue comme une grille discrète.

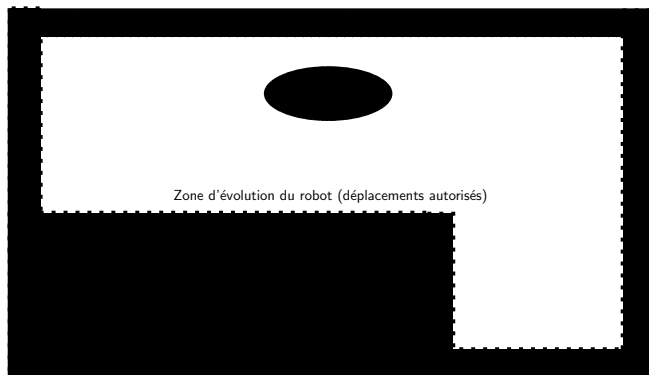
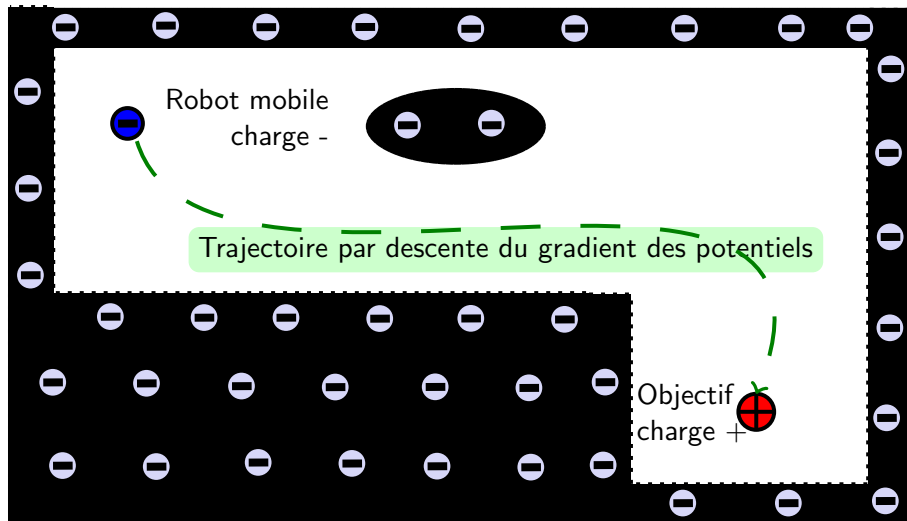


Figure – Carte d'occupation obtenue après dilatation.

# Méthode des potentiels I

Illustration : le robot et les obstacles se repoussent, l'objectif attire le robot.





# Méthode des potentiels II

Ainsi, on peut définir les potentiels fictifs :

## Potentiel attracteur

On utilisera la forme quadratique suivante autour du point objectif final  $x_f, y_f$ .

$$U_a(x, y) = \frac{1}{2}\lambda[(x - x_f)^2 + (y - y_f)^2]$$

$\lambda$  est un paramètre servant à ajuster la pondération du potentiel attracteur, il faut que ce dernier ait une influence significative sur entre  $x_0, y_0$  et  $x_f, y_f$ . Il est nul sur le point objectif et croît en  $d(x, y)^2$  d'autant plus que l'on s'éloigne dudit point. Son gradient vaut :

$$\nabla U_a = \nabla U_{ATR} = \lambda \begin{bmatrix} x - x_f \\ y - y_f \end{bmatrix},$$

et a le bon goût de rester continu.

## Potentiel répulsif

$$U_e(x, y) = \begin{cases} \frac{\mu}{2} \left( \frac{1}{d(x, y)} - \frac{1}{\rho_0} \right)^2 & \text{si } (x - x_e)^2 + (y - y_e)^2 = d^2(x, y) < \rho_0^2 \\ 0 & \text{sinon} \end{cases}$$

Le gradient vaut :

$$\nabla_e(x, y) = \begin{cases} \frac{-\mu \left( \frac{1}{d(x, y)} - \frac{1}{\rho_0} \right)}{d^3(x, y)} \begin{bmatrix} (x - x_e) \\ (y - y_e) \end{bmatrix} & \text{si } d(x, y) < \rho_0 \\ 0 & \text{sinon} \end{cases}$$

Cette fonction quadratique ne comporte pas de discontinuités pour son gradient, ce qui est un atout pour ne pas avoir numériquement des changements brusques.

Le schéma de descente de gradient peut être donné comme suit :

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} - \alpha_i \begin{bmatrix} \nabla_U^x(x_i, y_i) \\ \nabla_U^y(x_i, y_i) \end{bmatrix}$$

on trouvera le minimum global si la somme des potentiel est convexe. Comprendre l'objectif final.

Mais plusieurs obstacles se dressent devant nous :

- La carte des obstacles est discrète, une case représente  $n$  cm par  $n$  cm, le potentiel est le même dans chacune des cases ;
- Les valeurs des gradients sont fort différentes d'un point à un autre, avec un mauvais pas  $\alpha_i$  on peut se retrouver hors des limites du terrain !

## Solution :

- Considérer un gradient normalisé qui donne un déplacement constant à chaque itération ;
- Un pas constant  $\alpha > 1$  de valeur modérée (on a choisi)  $\alpha = \sqrt{2}$  dans nos simulations ;
- Considérer la partie entière (notée  $\mathcal{E}[\bullet]$ ) de ce pas normalisé avec  $x_i, y_i$  entiers représentant les coordonnées de la grille.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} - \mathcal{E} \left( \frac{\alpha}{\sqrt{[\nabla_U^x(x_i, y_i)]^2 + [\nabla_U^y(x_i, y_i)]^2}} \begin{bmatrix} \nabla_U^x(x_i, y_i) \\ \nabla_U^y(x_i, y_i) \end{bmatrix} \right)$$

# Lissage des courbes

On peut moyenner les points trouvés par la descente de gradient pour rendre la trajectoire plus lisse :

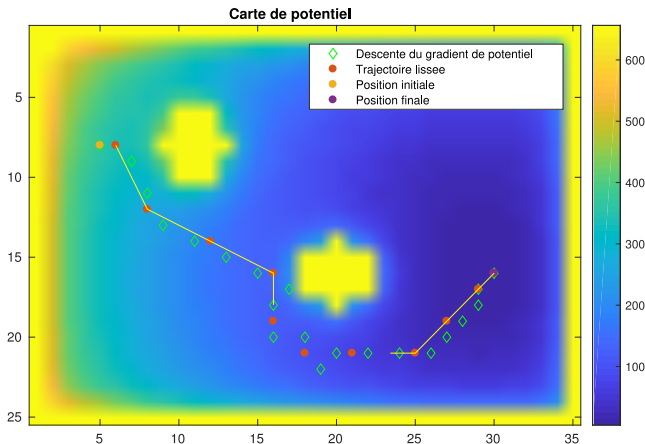


Figure – Exemple de descente de gradient lissée.

Regardons-le et exécutons un exemple.

Le script Matlab `planification.m` dans le répertoire `chemins` montre l'optimisation du gradient.