Cinématique d'un robot à montage différentiel et algorithme de suivi de chemin

Justin Cano,
E-Gab Centrale Marseille, promo entrante 2014
cano.justin@gmail.com

9 janvier 2018

Table des matières

1 Avant-propos **1**

2 Cinématique **1**
2.1 Hypothèses et notations 1
2.2 État cinématique du robot 2
2.3 Calcul de l'état en fonction des vitesses de rotation des roues 2
2.4 Équation dans l'espace de travail et intégration des états 2

3 Contrôleur point à point **4**
3.1 Définition du contrôleur de position point à point 4
3.2 Définition du contrôleur d'orientation 5
3.3 Quand est atteint l'objectif ? 6

4 Planification de trajectoire **7**
4.1 Carte 2D discrétisée des obstacles 7
4.2 Méthode des potentiels et descente du gradient 8
4.2.1 Potentiel attracteur 9
4.2.2 Potentiel répulsif 9
4.2.3 Potentiels répulsifs, calculons les vite ! 10
4.2.4 Carte des gradients de potentiels totaux 10
4.3 Un simple algorithme de planification 11
4.4 Comment intégrer de nouveaux obstacles (ex : robot) ? 12
4.5 Minima locaux, ≪pièges≫ de potentiel 12

Table des figures

1 Cinématique du robot différentiel (vue de dessus) 1
2 Illustration d'intégration des mesures d'un profil de vitesse trapézoïdal biaisées et entachées d'un léger (DSP : 0.1) bruit gaussien. 3
3 Architecture générale pour le contrôleur point à point. 5
4 Sélection du contrôleur. 6
5 Environnement connu. 7
6 Environnement dilaté. 7
7 Carte d'occupation obtenue après dilatation. 8
8 Illustration du principe des charges fictives. 8
9 Exemple de carte d'obstacle dilatée sous `Matlab`. 9
10 Allure des potentiels calculés sous `Matlab` pour la carte d'obstacle de la figure 9. . . . 10
11 Carte des orientations et des magnitudes des gradients de potentiel illustrant le problème de régularisation. 11
12 Résultat finalement obtenu pour l'exemple de la figure 9. 13

1 Avant-propos

Ce document traite du suivi de chemin d'une base roulante à montage différentiel, nous supposerons que la vitesse de rotation des roues est asservie. Ayant écrit un autre mémo technique sur le sujet, ce dernier se trouve sur le site suivant http://justincano.com. Ce document est assorti de scripts Matlab pour mieux le comprendre situés sur le même site.

Résumé de la démarche : On veut atteindre depuis le point $A = (x, y, \theta)$ le point $B = (x', y', \theta')$. La démarche intuitive est de planifier une trajectoire, c'est à dire un ensemble de points intermédiaires, évitant les obstacles au maximum. Il faut suivre ces derniers de manière la plus linéaire possible, sans à coups (afin de ne pas endommager le robot et ses moteurs). Mais avant de faire ceci, il faut se donner un modèle de robot et calculer sa cinématique (étude des vitesses de déplacement) afin de comprendre comment le commander. Le tout en essayant de maîtriser les perturbations et les changements de topologie du terrain.

2 Cinématique

2.1 Hypothèses et notations

Le robot est équilibré et son centre de masse se situe au milieu du segment qui relie les centres des deux roues. Nous noterons ce centre M, les coordonnées du robot dans l'espace de travail sont celles du centre de masse et sont notées (x, y). On suppose que l'origine de l'espace de travail est connue et correspond à $O = (0, 0)$.

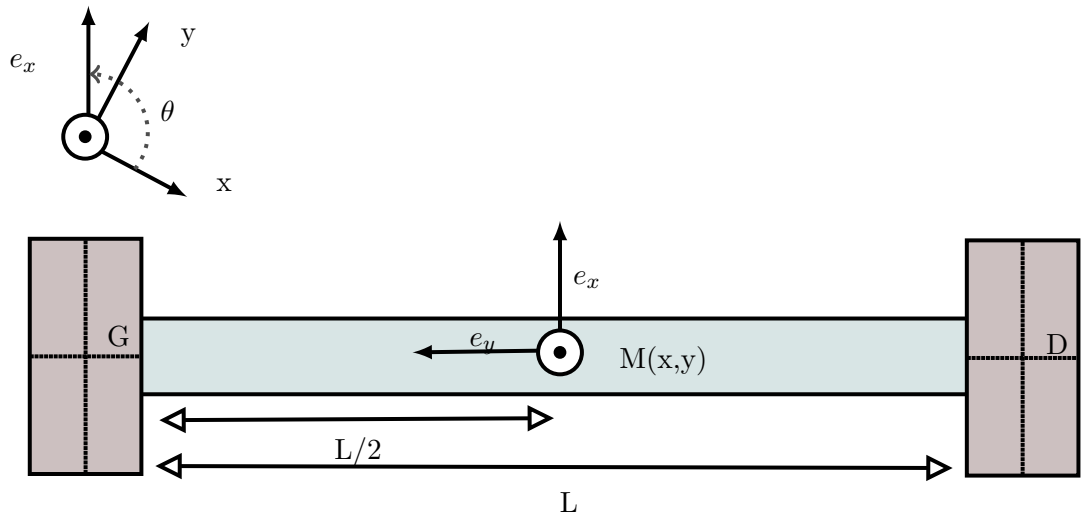

FIGURE 1 – Cinématique du robot différentiel (vue de dessus)

On peut supposer que le robot roule sans glisser. Ce qui nous permet de dire que la vitesse $\vec{v}$ tangentielle du point M, par rapport au référentiel de l'espace de travail (O, x, y) que nous allons appeler vitesse tangentielle du robot n'a pas de composante suivant e_y.

$$\vec{v}.\vec{e_y} = 0$$

Chacune des roues dispose d'un moteur qui peut l'entraîner à la vitesse ω_g ou ω_d. Nous noterons que ces vitesses de rotation sont selon un angle positif et selon l'axe e_y, et il qu'il faut faire attention au signe. Pour des raisons de clarté dans la démarche, nous partirons des définitions des vitesses tangentielles des roues (en notant R_k les rayons de ces dernières) en supposant qu'elles sont positives selon l'axe tangentiel e_x du robot [1] :

$$\vec{v_g} = \omega_g R_g \vec{e_x} \qquad \vec{v_d} = \omega_d R_d \vec{e_x}$$

1. Attention à bien implémenter le signe des vitesses ω_k en pratique !

2.2 État cinématique du robot

On tâchera de déterminer deux vitesses pour le robot afin de décrire son état cinématique :

— Une vitesse de rotation du robot qui est en fait la dérivée de l'angle θ, nous noterons $\dot{\theta} = \Omega$ cette dernière. Notons qu'elle se trouve selon l'axe e_z dans les référentiels de l'espace de travail (fixe) et du robot (mobile).

— Une vitesse tangentielle $\vec{V} = V\vec{e_x}$ qui s'exprime dans le référentiel de l'espace de travail ainsi $\vec{V} = \dot{x}\vec{x} + \dot{y}\vec{y}$.

La représentation d'état $\boldsymbol{X}$ cinématique du robot, exprimée dans le référentiel fixe (O, x, y) peut ainsi s'écrire :

$$\frac{d\boldsymbol{X}}{dt} = \frac{d}{dt}\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \Omega \end{bmatrix}$$

Il s'agit d'un formalisme important pour le reste de notre étude. En fait, on désire se déplacer et s'orienter dans le plan, si on peut obtenir un état x, y, θ quelconque, on contrôle parfaitement la base roulante du robot. Ce que l'on tâche de faire durant cette formation en fait.

Pourquoi deux vitesses de rotation donnent trois états ? La réponse vient d'une contrainte de roulement sans glissement, appelée contrainte non-holonome en mécanique du solide. Contrainte que nous avons vu à la section précédente et qui impose une vitesse nulle selon e_y.

2.3 Calcul de l'état en fonction des vitesses de rotation des roues

Tout d'abord, définissons les vitesses aux centres de rotation G et D des roues gauche et droite de deux façons différentes [2] :

$$\vec{V_g} = \vec{V_M} + \vec{\Omega} \times \vec{GM} = V\vec{e_x} + \Omega\vec{e_z} \times (-\frac{L}{2}\vec{e_y}) = (V + \frac{\Omega L}{2})\vec{e_x} = \omega_g R_g \vec{e_x}$$
$$\vec{V_d} = \vec{V_M} + \vec{\Omega} \times \vec{DM} = V\vec{e_x} + \Omega\vec{e_z} \times (+\frac{L}{2}\vec{e_y}) = (V - \frac{\Omega L}{2})\vec{e_x} = \omega_d R_d \vec{e_x}$$

Après projection sur $\vec{e_x}$ On a donc un système de deux équations de deux inconnues que l'on peut résoudre. Pour la vitesse de translation suivant e_x , on a :

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

Et pour la vitesse de rotation, nous avons :

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

Remarque : Je note $\times$ le produit vectoriel (conventions nord-américaines).

2.4 Équation dans l'espace de travail et intégration des états

Afin d'obtenir l'état $\boldsymbol{X}$, on doit appliquer une projection sur les axes de la vitesse tangentielle décrivant un angle θ ainsi on a les équations suivantes :

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

On peut donc résoudre en tout temps l'équation :

$$\boldsymbol{X}(T) = \int_{t=t_0}^{T} \begin{bmatrix} V\cos(\theta) \\ V\sin(\theta) \\ \Omega \end{bmatrix} + \boldsymbol{X}(t_0)$$

2. Exprimées dans le référentiel (M, e_x, e_y) du robot.

Mais en pratique, si la mesure de vitesse est bruitée ou est victime d'un biais, un terme d'erreur aléatoire $\boldsymbol{\epsilon}$ apparaît :

$$\tilde{\boldsymbol{X}} = \boldsymbol{X} + \boldsymbol{\epsilon}$$

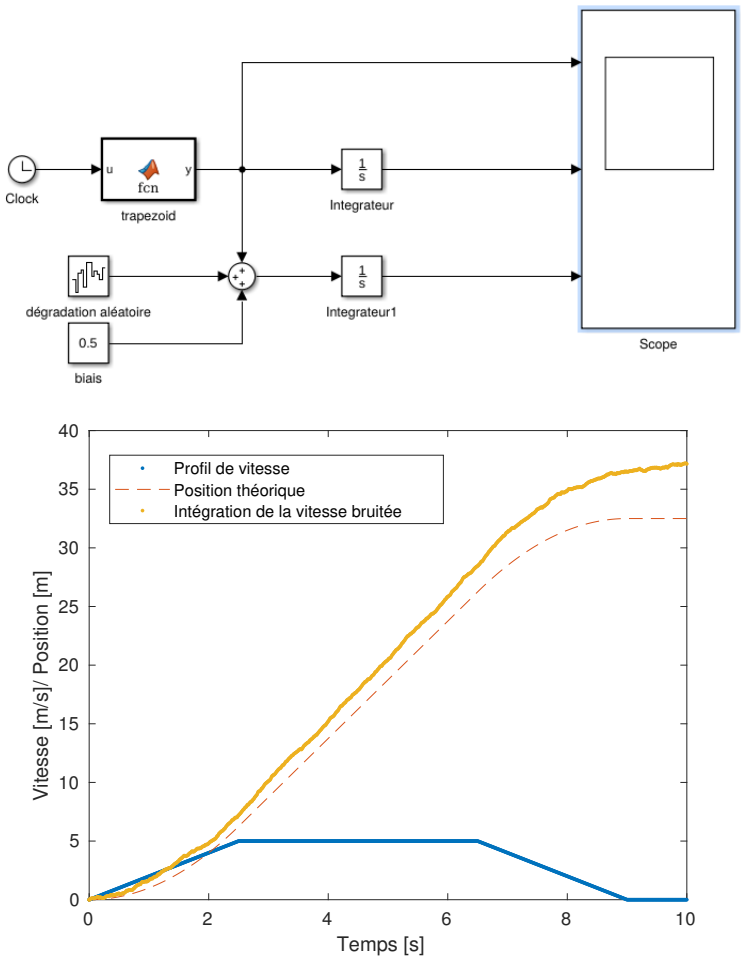

FIGURE 2 – Illustration d'intégration des mesures d'un profil de vitesse trapézoïdal biaisées et entachées d'un léger (DSP : 0.1) bruit gaussien.
Ce profil est volontairement entaché d'un grand biais pour illustrer le phénomène à long terme.

L'intégration de ce terme perturbateur sur un trop gros temps risque de donner une mesure trop entachée d'erreurs (exemple, figure 2). Il faut donc disposer d'information de positionnement externes en plus. Dans la littérature un système intégrant des équations de navigation s'appelle un système de *dead reckonning* (ex : système inertiel, intégration de l'odométrie [3]), pour l'aider à obtenir des données de recalage on l'adjoint d'un système de *position fixing*. On peut citer les systèmes de type GPS, des balises qui viennent donner des mesures directement dans l'espace de travail ou encore un système de caméra fixe filmant le robot évoluant dans son plan de haut.

Comment estimer x et y en pratique ? On peut tout simplement utiliser la mesure des encodeurs mais ceci est valide sous de faibles distances seulement (voir remarque précédente). Il suffit de nous rappeler que l'on reçoit chaque δt un tick de l'encodeur correspondant à un angle $\delta\omega^i$ des roues. Ainsi dans l'espace de travail, nous avons :

$$\tilde{x} = \int_{t=0}^{T=N\delta T} V\cos(\theta)dt \approx \sum_{i\in[0,N\delta t]} \frac{\delta\omega_g^i R_g + \delta\omega_d^i R_g}{2}\cos(\theta^i)$$

3. Comme ici ou on intègre des vitesses estimées par les codeurs incrémentaux des roues.

Et respectivement :

$$\tilde{y} = \int_{t=0}^{T=N\delta T} V \sin(\theta) dt \approx \sum_{i \in [0, N\delta t]} \frac{\delta\omega_g^i R_g + \delta\omega_d^i R_g}{2} \sin(\theta^i)$$

La question est comment estimer thêta alors ?

Notes sur l'angle θ : En ce qui concerne ce dernier, il nous est possible de le mesurer directement avec l'aide d'un magnétomètre trois axes, dans une zone de travail restreinte, on peut supposer que le vecteur magnétique est constant dans cette zone au vu de la superficie de la Terre :

$$\boldsymbol{m}(x, y) \approx \boldsymbol{m} = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}$$

Si on mesure le champ magnétique terrestre lorsque le robot est orienté en direction $\vec{x}$. $\theta_m^x = (\vec{x}, \begin{bmatrix} m_x & m_y \end{bmatrix}^T) = \texttt{atan2}(m_y, m_x)$, alors on est capable de déduire θ en tout point de l'espace de travail :

$$\tilde{\theta} = \texttt{atan2}(m_y, m_x) - \theta_m^x$$

On supposera le magnétomètre comme étant à plat c'est à dire que la mesure de la composante verticale du champ magnétique peut être ignorée, le problème étant bidimensionnel.

3 Contrôleur point à point

Maintenant que nous connaissons notre cinématique, nous allons donner au robot une information pour se déplacer dans l'espace d'état. C'est-à-dire d'une position (x_i, y_i) initiale à une position (x_d, y_d) désirée ET d'une orientation θ_i à une orientation θ_d. En somme on demande des échelons de consigne comme classiquement en automatique. On considérera, pour des raisons d'évitement d'obstacle et de risque de dépassement relatifs trop conséquents que les distances entre les états $\boldsymbol{X}_i$ et $\boldsymbol{X}_f$ sont faibles, nous verrons à la section suivante comment définir une suite d'états intermédiaires sur des ≪grandes distances≫ dans la carte.

3.1 Définition du contrôleur de position point à point

L'idée est ici de nous ramener dans un cas polaire [Astolfi, 1999, Gourdeau, 2017]. On peut définir la distance euclidienne au point désiré :

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

L'angle distant entre ≪le cap≫ du robot et le point désiré :

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

L'angle entre ≪le cap≫ et le vecteur $\vec{x}$:

$$\beta = -\theta - \alpha$$

On pourra donc boucler (on enverra ces commandes aux roues avec les relations cinématiques trouvées précédemment) le système avec trois gains k_d, k_α, k_β :

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

Notons que la fonction `atan2` dernière fonctionne comme $\text{atan}(y/x)$ renvoyant des valeurs inclues dans $[-\pi, \pi]$, elle est codée en C/C++ , Python, Matlab...

Stabilité locale : Notons que nous devons respecter les trois relations suivantes pour obtenir un contrôleur localement stable autour du point d'équilibre [Gourdeau, 2017, Astolfi, 1999] :

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

Ce résultat s'obtient en linéarisant le système cinématique autour d'un point d'équilibre. On peut trouver les inégalités en prétendant que les pôles, valeurs propres de la matrice dynamique linéarisée ont une partie réelle négative (pôles stables, localement). Le terme local renvoie au fait que le système est approché autour d'un point d'équilibre.

3.2 Définition du contrôleur d'orientation

Une fois l'objectif en position atteint, on peut utiliser un simple contrôleur pour s'orienter comme voulu en faisant un bouclage proportionnel sur la commande en vitesse de rotation :

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

Notons qu'il faut faire attention aux modulos lors de l'implémentation, ceci faisant changer le signe de la commande proportionnelle.

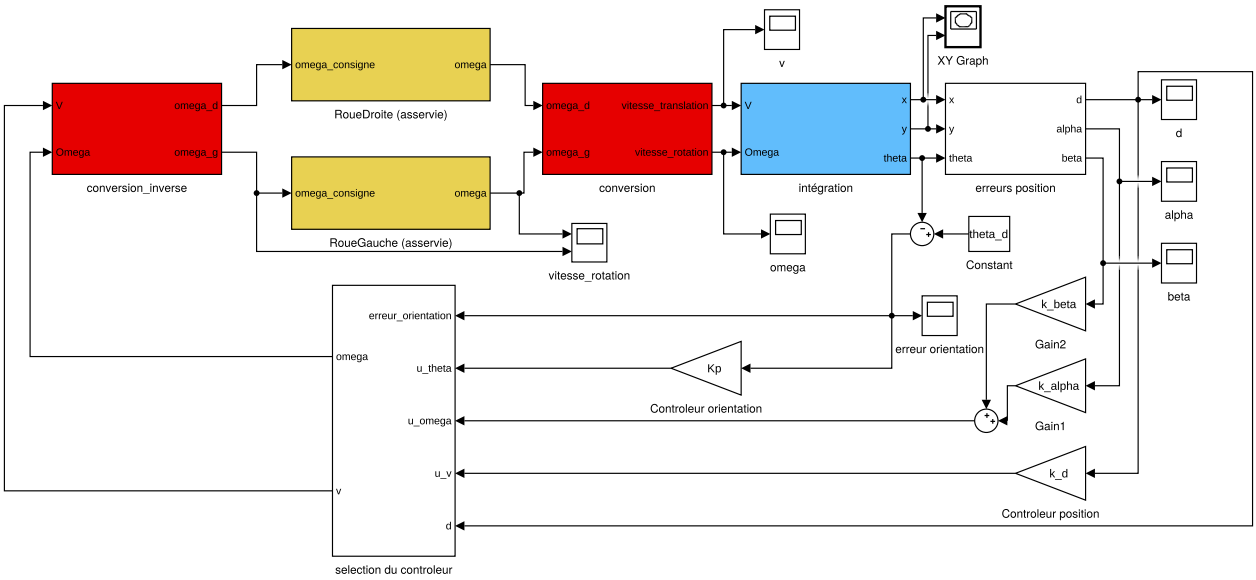

FIGURE 3 – Architecture générale pour le contrôleur point à point.

3.3 Quand est atteint l'objectif ?

On peut arrêter le contrôleur tangentiel lorsque nous nous trouvons à une distance inférieure à un certain seuil ρ de l'objectif (x_d, y_d). Sinon, nous risquons de tourner autour de notre objectif pour peu de gain de précision.

$$V(t) = \begin{cases} 0, \text{ si } (x - x_d)^2 + (y - y_d)^2 < \rho^2 \\ c(t) \star \epsilon_t(t) \text{ sinon} \end{cases}$$

Ensuite, on peut faire de même pour la vitesse angulaire et un seuil γ, mais une fois le premier seuil atteint seulement (risque de cap identique lors du déplacement [4]) :

$$\Omega(t) = \begin{cases} 0, \text{ si } (\theta - \theta_D)^2 < \gamma^2 \textbf{ ET si}(x - x_d)^2 + (y - y_d)^2 < \rho^2 \\ c(t) \star \epsilon_\Omega(t) \text{ sinon} \end{cases}$$

En pratique, on désignera un nouvel objectif $\boldsymbol{X}_d^{i+1}$ dès que $\boldsymbol{X}_d^i$ est atteint (ie lorsque les deux tests sont vrais). Si on est au dernier objectif du chemin planifié, on peut être plus exigeant sur les critères d'arrêt. On peut aussi n'appliquer le contrôleur d'orientation qu'au dernier point si le chemin suivi est suffisamment rectiligne.

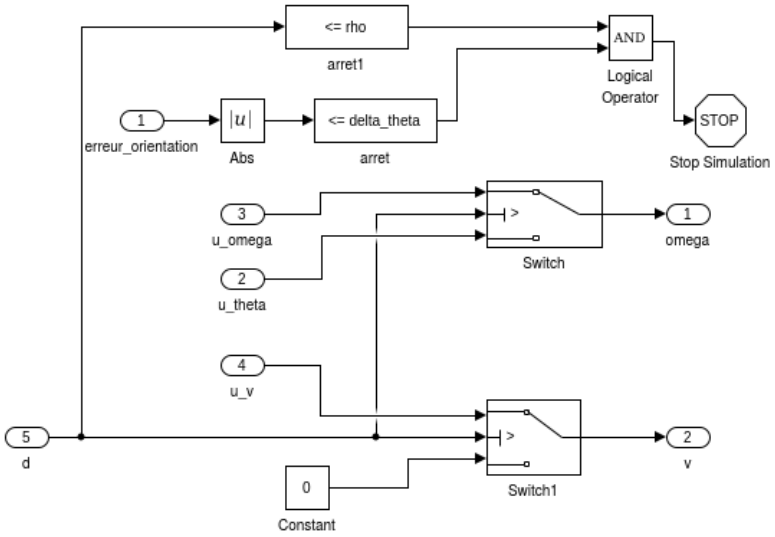

FIGURE 4 – Sélection du contrôleur.

4. C'est pour ceci que ce contrôleur doit être plus lent que le tangentiel...

4 Planification de trajectoire

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

$$X_d^1, X_d^2, ..., X_d^{N-1}$$

De manière automatisée pour nous rendre de X_d^0 à X_d^N qui sont la position initiale du robot et la position finale voulue.

4.1 Carte 2D discrétisée des obstacles

Soit un environnement connu avec des obstacles (figure 7).

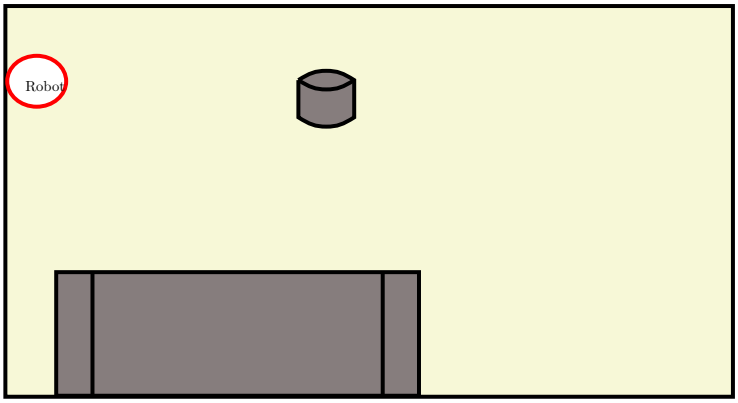

FIGURE 5 – Environnement connu.

On va découper tout d'abord la carte en pixels de H centimètres de côtés. Chaque pixel $P_{i,j}$ dispose d'un centre $C_{i,j}$ se trouvant au milieu de ce dernier. Chaque pixel occupé par un obstacle doit être mis à 1, les autres restent à zéro. On appelle cette carte la carte d'occupation.

Le robot est inscrit dans un cercle de rayon de R centimètres. On pose le nombre de pixels à dilater $\Delta = E^+(R/H)$ où E^+ est la fonction partie entière arrondissant au nombre entier supérieur.

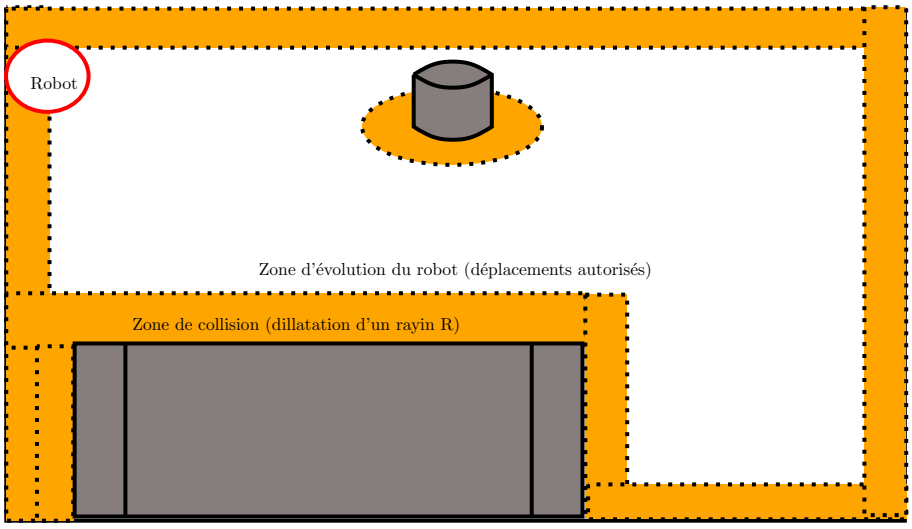

FIGURE 6 – Environnement dilaté.

On applique l'opération de dilatation (illustrée à la figure 6) avec pour forme un carré de Δ de côté sur la carte, ceci nous assure que le robot pourra passer :

$$\mathbf{DIL}[\square_{Delta,i,j}](Pi,j) = \begin{cases} 1 \text{ si } \exists P_{i',j'} = 1 \in \square_{Delta,i,j} \\ 0 \text{ sinon} \end{cases}$$

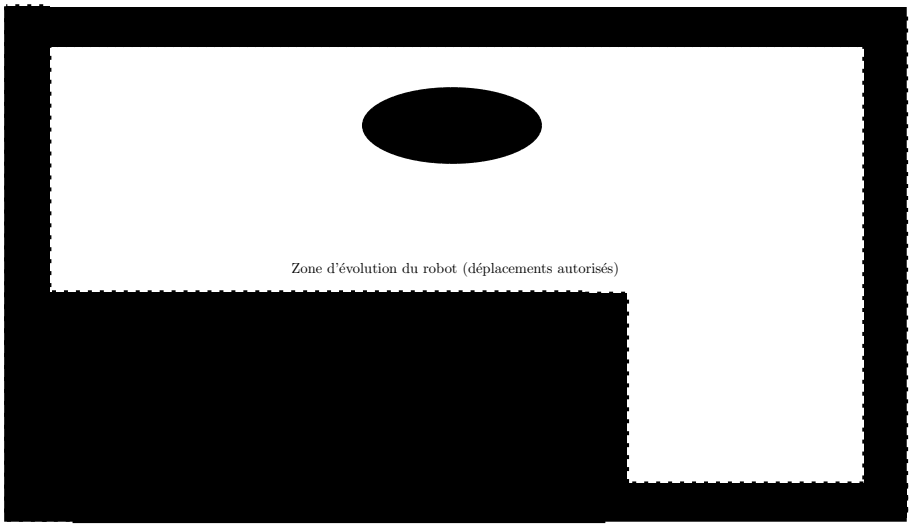

FIGURE 7 – Carte d'occupation obtenue après dilatation.

4.2 Méthode des potentiels et descente du gradient

ATTENTION : Cette méthode ne planifie que les positions et non les orientations. Il faut faire attention de bien pivoter le robot sur lui même près de l'objectif si il doit être complètement retourné par rapport au sens du gradient.
On suppose que le robot est une charge négative libre (on va dire un électron). Naturellement, on sait que notre robot doit converger vers un point (x_f, y_f) on va dire que cette charge est une charge positive fixe, le robot sera attiré par elle. Les obstacles quant à eux sont des charges à potentiel limité (répulsion limitée) négatives. On cherchera donc à minimiser le potentiel, à descendre en potentiel, sur la carte ainsi modifiée.

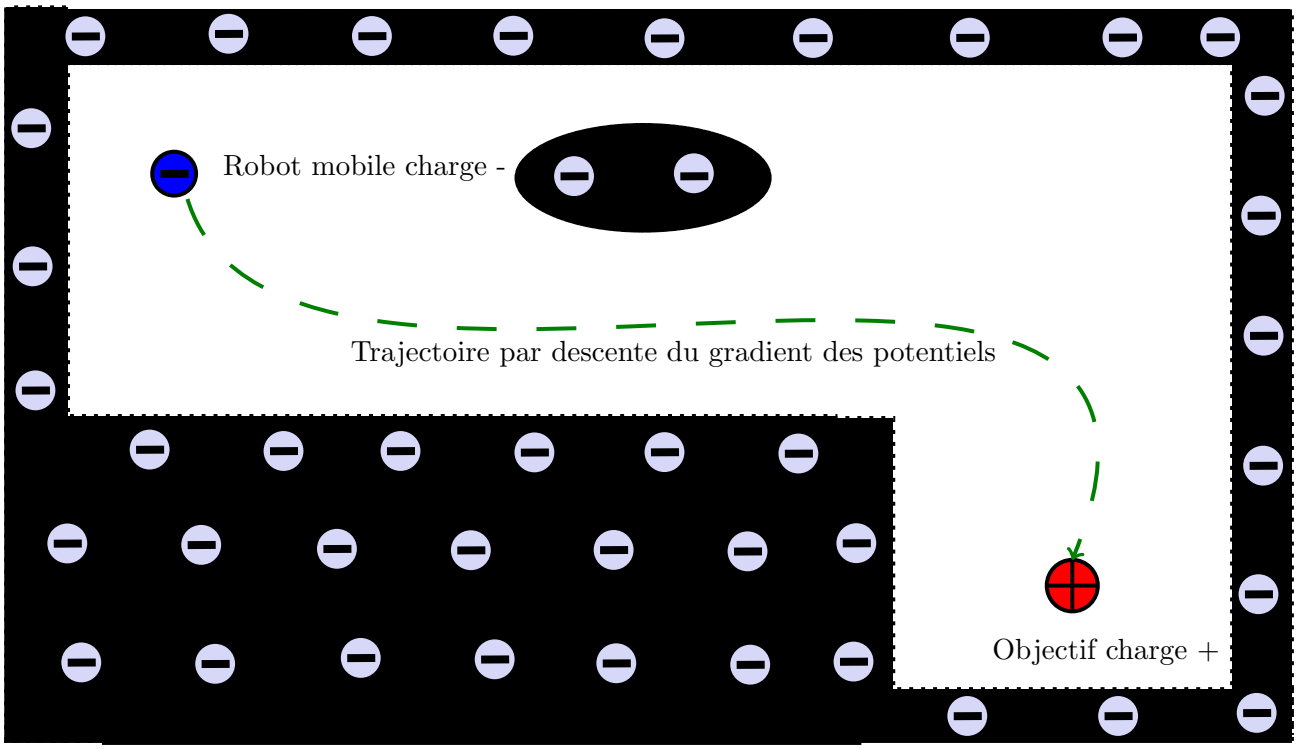

FIGURE 8 – Illustration du principe des charges fictives.

4.2.1 Potentiel attracteur

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

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

λ 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 croit en $d(x,y)^2$ d'autant plus que l'on s'éloigne dudit point. Son gradient vaut :

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

4.2.2 Potentiel répulsif

On utilisera la forme quadratique inverse autour de l'obstacle à éviter x_e, y_e. Ces potentiels ont une influence autour du point à éviter [5], plus précisément dans un disque de rayon $\rho_0 < d(x,y)$.
$d(x,y) = \sqrt{(x-x_e)^2 + (y-y_e)^2}$ étant la distance euclidienne à l'obstacle considéré.

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

Le gradient vaut :

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

Cette fonction quadratique ne comporte pas de discontinuités pour son gradient, ce qui est un atout pour ne pas avoir numériquement des changements brusques. **Notons que chaque pixel occupé de la carte génère un champ répulsif**.

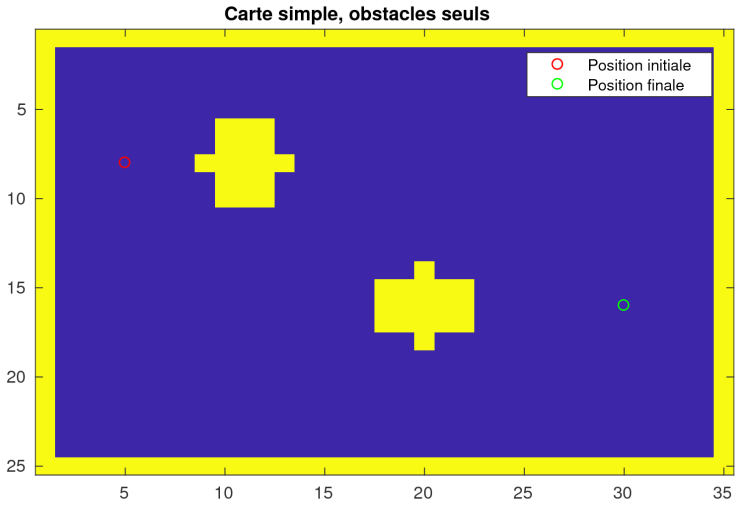

FIGURE 9 – Exemple de carte d'obstacle dilatée sous `Matlab`.

5. On remarquera que l'infini est un potentiel possible, quelques langages de programmation le gèrent automatiquement mais je vous conseille de faire un test et de mettre au point obstacle une valeur infinie sans perdre du temps à la calculer pour avoir potentiellement une erreur par la suite (on peut obtenir une valeur erreur NAN, Not A Number, alors que la valeur INF peut exister, sinon prendre une valeur considérablement élevée).

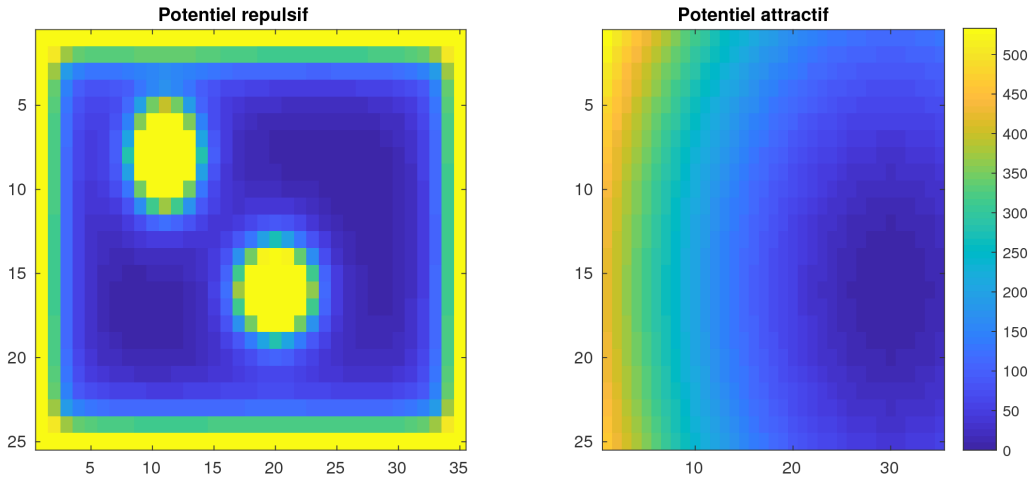

FIGURE 10 – Allure des potentiels calculés sous `Matlab` pour la carte d'obstacle de la figure 9.

4.2.3 Potentiels répulsifs, calculons les vite !

Oui, un robot cherche à vite calculer ses potentiels, il existe une méthode très simple pour ça : le masque additif. Pour chacune des cartes de gradient (matrice représentant le gradient composante x ou y sur le terrain cartographié), on peut créer un masque additif. Par exemple, on prend $\lambda = 0.1$ et $\rho = 31$ cm. On suppose qu'on a découpé la carte en pixels de 10 cm de coté. On sait donc que l'influence du potentiel répulsif sera de trois cases de rayons autour de l'obstacle. Donc, un masque de `7x7` cases fera l'affaire et on peut le pré-calculer :

$$M^x_{\nabla e} = \begin{bmatrix} 0.0007992 & 0.0015687 & 0.002694 & 0.0033453 & 0.002694 & 0.0015687 & 0.0007992 \\ 0.0010458 & 0.0028399 & 0.007423 & 0.011694 & 0.007423 & 0.0028399 & 0.0010458 \\ 0.00089799 & 0.0037115 & 0.02386 & 0.096774 & 0.02386 & 0.0037115 & 0.00089799 \\ 0 & 0 & 0 & Inf & 0 & 0 & 0 \\ -0.00089799 & -0.0037115 & -0.02386 & -0.096774 & -0.02386 & -0.0037115 & -0.00089799 \\ -0.0010458 & -0.0028399 & -0.007423 & -0.011694 & -0.007423 & -0.0028399 & -0.0010458 \\ -0.0007992 & -0.0015687 & -0.002694 & -0.0033453 & -0.002694 & -0.0015687 & -0.0007992 \end{bmatrix}$$

Et de plus, $M^y_{\nabla e} = (M^x_{\nabla e})^T$ ce qui est normal puisque la fonction de potentiel est symétrique pour x et y.

On perd moins de temps, à chaque pixel de la carte d'occupation dilatée mis à 1, on ajoute 49 termes pour chaque composante du gradient. Au lieu d'avoir les multiplications en sus, ce qui prend du temps en point flottant. Comme on cherche la rapidité, ceci n'est pas optimal.

4.2.4 Carte des gradients de potentiels totaux

Il suffit de tout ajouter. Chaque pixel occupé de la carte d'occupation crée un potentiel répulsif. On peut commencer par faire la carte des gradients de potentiels répulsifs. Pour chaque pixel occupé, on va sommer les potentiels des deux matrices $M^y_{\nabla e}$ et $M^x_{\nabla e}$ dans les deux cartes correspondantes autour du pixel qui se verra attribuer la valeur interdite ∞. Si un pixel du voisinage n'appartient pas à la carte, ignorer l'opération (il faut implémenter un test lorsqu'on parcourt le voisinage de l'obstacle). Ainsi on obtient :

$$\nabla^x_{REP}(x,y) = \sum_{P(x,y)==1} \sum_{(i,j)\in \text{ voisinage } (x,y)} M^x_{\nabla e}(i,j)$$

Et respectivement pour la composante y. Puis, il suffit d'ajouter le gradient attracteur (calculé pour tout pixel non-occupé appartenant à la carte) au gradient du potentiel répulsif total pour obtenir le

gradient final :

$$\nabla_U(x,y) = \begin{bmatrix} \nabla^x_{REP}(x,y) + \nabla^x_{ATR}(x,y) \\ \nabla^y_{REP}(x,y) + \nabla^y_{ATR}(x,y) \end{bmatrix}$$

Tout ceci peut être pré-calculé pour une grande partie (la majorité des obstacles sont connus dans des compétitions de robotique par exemple).

4.3 Un simple algorithme de planification

Supposons que nous disposons de la carte précédemment calculée. Il suffit donc de «suivre» le gradient pour obtenir une trajectoire convenable qui ne heurte en aucun cas les obstacles. On va utiliser une approche, s'inspirant de la célébrissime méthode de descente de gradient pour faire ceci. Supposons que nous nous trouvons à un point de l'espace d'état i, déjà calculé ($i \in [1, N-2]$) ou initial $i = 0$:

$$\boldsymbol{X}_i = \begin{bmatrix} x_i & y_i & \theta_i \end{bmatrix}^T$$

On peut passer en $i+1$ aisément avec les formules suivantes. Tout d'abord, on calcule les coordonnées par méthode du gradient :

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} - \alpha_i \begin{bmatrix} \nabla^x_U(x_i, y_i) \\ \nabla^y_U(x_i, y_i) \end{bmatrix}$$

Il faut faire attention... les positions ne sont pas nécessairement des entiers (comme nous avons discrétisé notre carte) il faut arrondir à l'entier le plus proche les coordonnées. Cette remarque est cruciale pour l'implémentation. Évidemment α_i est un paramètre à ajuster, il faut faire attention à ce qu'il ne soit pas trop élevé sinon, on peut foncer dans des obstacles... ou à l'inverse rester bloqués par un trop petit gradient ! Voir figure 11...

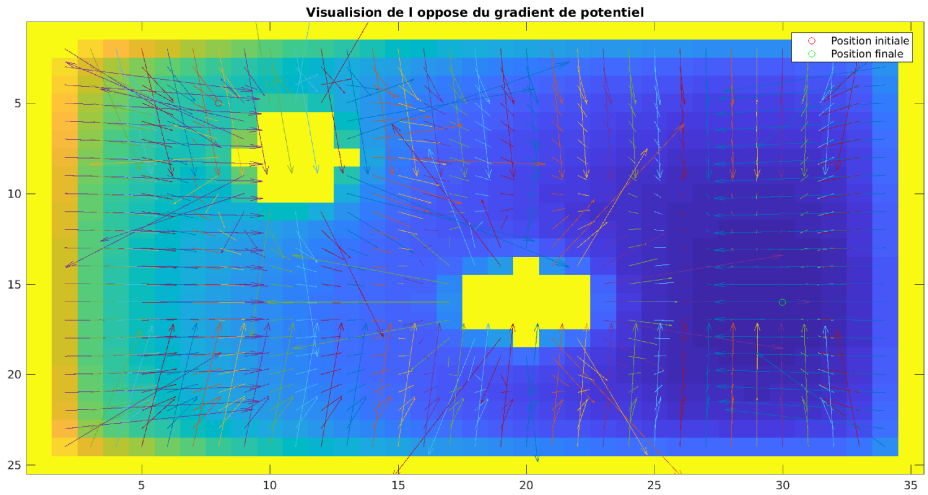

FIGURE 11 – Carte des orientations et des magnitudes des gradients de potentiel illustrant le problème de régularisation.

Comment résoudre ceci ? Tout simplement, il faut prendre une valeur normalisée du gradient total et la multiplier par une constante supérieure à 1 qui serait un rayon maximum entre deux déplacement. Ainsi, par exemple, si $\alpha_i = 1.41 \frac{1}{\sqrt{[\nabla^x_U(x_i,y_i)]^2 + [\nabla^y_U(x_i,y_i)]^2}}$ soit l'inverse de la norme du gradient, alors ce dernier sera un vecteur qui pointera sur un cercle de rayon $1.41 \approx \sqrt{2}$ autour du pixel concerné. Ainsi, nous avons pu obtenir un gradient qui amenait la position k du robot non loin de sa position $k+1$. Puis l'application de cette étape peut être écrite comme ceci avec $\mathcal{E}$ la fonction

partie entière à la valeur inférieure :

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} - \mathcal{E}(\frac{1.41}{\sqrt{[\nabla_U^x(x_i, y_i)]^2 + [\nabla_U^y(x_i, y_i)]^2}} \begin{bmatrix} \nabla_U^x(x_i, y_i) \\ \nabla_U^y(x_i, y_i) \end{bmatrix})$$

Ensuite (voir le script donné en annexe) il suffit de moyenner la courbe afin d'obtenir un résultat sans trop d'oscillations. Le temps de calcul est ici somme toute acceptable pour un ordinateur de bord type `Raspberry Pi` avant un trajet. Ainsi, nous avons déterminé une trajectoire à suivre de x_i, y_i. À nous de calculer les orientations maintenant.

Orientation ? Une fois ceci effectué, nous pouvons très simplement calculer l'orientation objectif (nous avons le gradient et la fonction `atan2`, donc c'est faisable !), qui suivra l'opposé du prochain gradient (direction opposées aux obstacles) :

$$\theta_{i+1} = \texttt{atan2}(-\nabla_U^y(x_{i+1}, y_{i+1}), -\nabla_U^x(x_{i+1}, y_{i+1}))$$

Rappelons nous que l'indice $i+1$ fait allusion au point de coordonnée **moyennée** extrait de l'algorithme précédent. Ce calcul peut être justifié si la topologie du terrain comporte des obstacles rapproché (risque de frapper ces derniers en parcourant un segment).

Et pour le dernier point ? Si on se situe à une distance raisonnable de ce dernier ($d(x, y) < \kappa$, seuil à déterminer selon la topologie du terrain). On peut tout simplement mettre en objectif pour le contrôleur point à point le point N qui est en fait l'état $\boldsymbol{X}_f$. Ceci évite de faire «tourner» le robot autour du point objectif en arrêtant la planification. Cependant, il faut faire attention à ce que l'orientation du gradient à l'étape $N-1$ ne soit pas trop éloignée de θ_f, si elle a lieu, un changement brusque pouvant occasionner des erreurs.

4.4 Comment intégrer de nouveaux obstacles (ex : robot) ?

Il suffit d'ajouter un nouveau champ répulsif dans ∇_{REP}^x ou ∇_{REP}^y et recalculer le trajet si besoin. Si ce dernier est mobile une étape de vérification peut être faite afin de savoir si le champ répulsif a toujours lieu d'être. Si ce n'est plus le cas, il peut être soustrait.

4.5 Minima locaux, «pièges» de potentiel

Il faut paramétrer les potentiels de manière à ce qu'il n'y en ait pas. Il faut faire également attention à planifier correctement les trajets objectif pour pas envoyer le robot dans un minimum de potentiel, typiquement envoyer le robot face à un obstacle hémicycle à l'instar d'un amphithéâtre et planifier un objectif derrière ce dernier... le robot restera coincé au centre de hémicycle... le risque est là, donc il faut tâcher de le minimiser lui aussi !.

Références

[Astolfi, 1999] Astolfi, A. (1999). Exponential stabilization of a wheeled mobile robot via discontinuous control. *TRANSACTIONS-AMERICAN SOCIETY OF MECHANICAL ENGINEERS JOURNAL OF DYNAMIC SYSTEMS MEASUREMENT AND CONTROL*, 121 :121–125.

[Gourdeau, 2017] Gourdeau, R. (2017). Cours de robotique. *Polytechnique Montréal*.

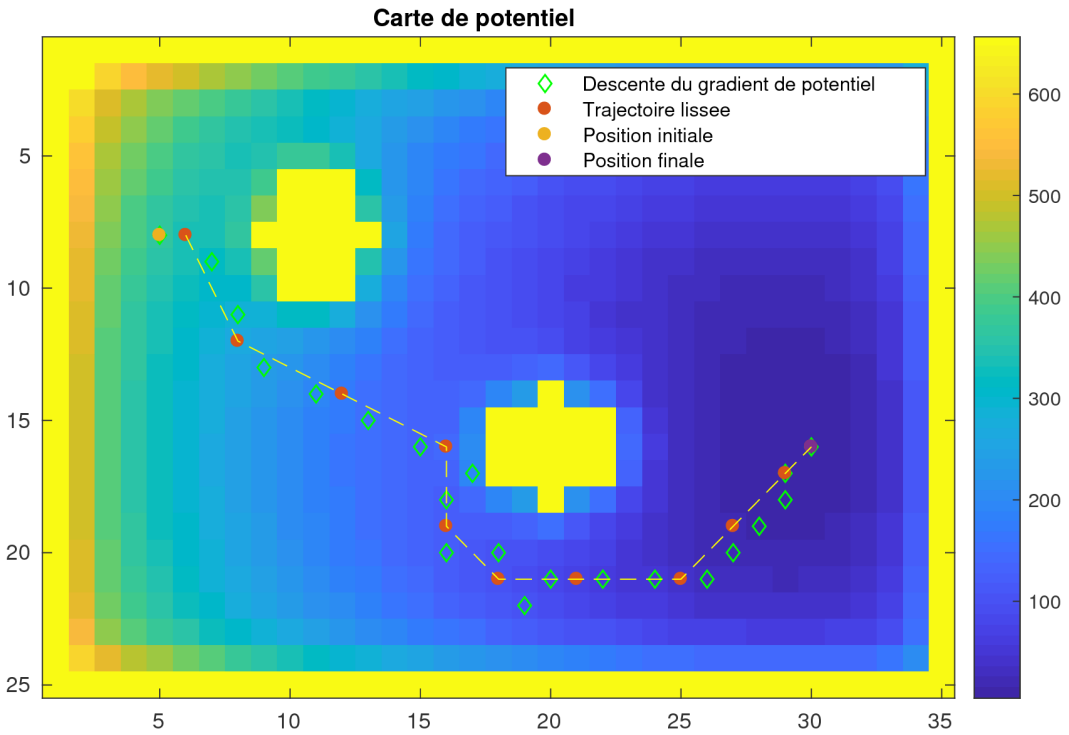

FIGURE 12 – Résultat finalement obtenu pour l'exemple de la figure 9.