Planification de mouvement pour un robot mobile
Méthode des champs de potentiels

PARTIE THEORIQUE :

A - Structure générale

Dans la planification de trajectoires de robot, le robot est considéré dans l'espace de configuration C comme une particule soumise à un **champ potentiel artificiel U(q)**, avec q l'état du robot (typiquement, pour un robot mobile q=(x,y)). A chaque itération, la **force artificielle F(q) = - ∇U(q)** induite par le champ potentiel, indiquera alors la direction la plus « prometteuse ».

$$F(q) = - \nabla U(q) = (\partial U/\partial x \quad \partial U/\partial y)^T \quad \text{, avec } q=(x,y)$$

Le champ potentiel est défini comme la somme d'un champ potentiel attracteur $U_{att}(q)$, poussant le robot vers la configuration finale, et un champ répulsif $U_{rep}(q)$, tenant le robot éloigné des obstacles.

$$U(q) = U_{att}(q) + U_{rep}(q)$$

B - Potentiel attractif

Le champ attractif peut être simplement défini sous la forme parabolique :

$$U_{att}(q) = ½ \, \xi \, \rho^2_{goal}(q)$$

avec ξ un scalaire positif et $\rho_{goal}(q)$ la distance euclidienne : $||q-q_{goal}||$. La fonction Uatt(q) est donc positive ou nulle, et atteint son minimum à q_{goal} , où $U_{att}(q_{goal})=0$.

La force F_{att} est différentiable partout dans C et on a :

$$F_{att}(q) = - \nabla U_{att}(q) = - \xi \, \rho_{goal}(q) \, \nabla \rho_{goal}(q) = - \xi \, (q-q_{goal})$$

Une autre forme de U_{att} peut être de type conique :

$$U_{att}(q) = \xi \, \rho_{goal}(q)$$

On aura alors une force de type :

$$F_{att}(q) = - \xi \, \nabla \rho_{goal}(q) = - \xi \, (q-q_{goal}) / \| q-q_{goal} \|$$

L'avantage de la forme conique sur la parabolique, c'est que la force est constante sur l'espace : elle ne tends pas vers l'infini lorsqu'on s'éloigne de q_{goal} comme la parabolique. Cependant, elle n'est pas nulle à q_{goal}.

C - Potentiel répulsif

Le potentiel répulsif doit servir à créer une **barrière de potentiel** autour des obstacles qui ne peut être traversé par le robot. De plus, on ne veut pas que ce potentiel affecte le mouvement du robot lorsque celui-ci est suffisamment loin des obstacles. Nous allons considérer deux types de potentiel répulsifs : exponentiel et hyperbolique.

Nous noterons $\rho(q) = \| q-q_{obst} \|$, la distance entre le robot et l'obstacle.

- **La forme exponentielle** est donnée par :

$$U_{rep}(q) = \eta \exp(- \rho(q) / \eta)$$

On a alors :

$$F_{rep}(q) = - \nabla U_{rep}(q)$$
$$= \exp(- \rho(q) / \eta) \nabla \rho(q)$$

Remarquons que l'influence de l'obstacle est non-nulle sur l'ensemble de l'espace (mais faible, voire négligeable en présence d'un champ attractif, à des relativement grandes distances).

- **La forme hyperbolique** est donnée par :

$$U_{rep}(q) = \tfrac{1}{2} \eta (1/\rho(q) - 1/\rho_0)^2 \text{ si } \rho(q) < \rho_0$$
$$= 0 \qquad \text{si } \rho(q) > \rho_0$$

ρ_0 est appelée la **distance d'influence** de l'obstacle. La fonction U_{rep} est positive ou nulle et tends vers l'infini lorsqu'on se rapproche de la frontière de l'obstacle.

On a alors :

$$F_{rep}(q) = - \nabla U_{rep}(q)$$
$$= \eta (1/\rho(q) - 1/\rho_0) 1/\rho^2(q) \nabla \rho(q) \text{ si } \rho(q) < \rho_0$$
$$= 0 \text{ si } \rho(q) > \rho_0$$

D - Planification de trajectoires

Une fois définis les potentiels, il suffit d'effectuer une descente de gradients pour atteindre la configuration finale. La descente de gradient consiste simplement à suivre la direction indiquée par la force F, et à avancer dans cette direction d'un pas de longueur δ_i. Par exemple, avec q=(x,y), on aura :

$$x(q_{i+1}) = x(q_i) + \delta_i \, \partial U/\partial x(x,y)$$
$$y(q_{i+1}) = y(q_i) + \delta_i \, \partial U/\partial y(x,y)$$

L'algorithme s'arrêtera lorsqu'on sera à une distance de q_{goal} inférieure à δ_i, dans le cas où un chemin est déterminé. L'inconvénient de cette méthode est que l'on peut tomber sur un minimum local. Pour éviter cela, on préfèrera souvent l'algorithme fourni en annexe (On nous vous demandera pas de le coder).

PARTIE EXPERIMENTALE :

Dans la partie expérimentale, nous allons considérer un cas simple où l'espace de configuration est limité à 2 dimensions avec des obstacles ponctuels (réduits à des points).

On va considérer un espace de configuration de taille 100x100, où l'on va disposer des obstacles ponctuels. Dans le code fourni (`potentialfield.m`), nous définissons deux structures : s et r. Ces structures possèdent trois champs :

- x : correspond à la position du point (attracteur pour s et répulsif pour r)
- w : le poids (ξ pour l'attracteur et η pour le répulsif)
- k : le choix de la fonction heuristique (conique/parabolique pour l'attracteur, exponentielle/hyperbolique pour le répulsif)

La variable q représente la position courante du robot. Les fonctions *fieldplot* et *robotplot* gèrent les affichages.

1 – Calcul et visualisation des champs potentiels

Compléter le code fourni afin de visualiser les champs de potentiels relatifs aux exemples fournis :

- cas 1 : champ attractif de type conique,
- cas 2 : champ attractif de type parabolique,
- cas 3 : champ répulsif de type exponentiel,
- cas 4 : champ répulsif de type hyperbolique.

Créer à partir des exemples fournis de nouveaux cas combinant des champs attractif et répulsif de différents types.

2 – Calcul et visualisation du chemin : Descente de gradient

Nous allons maintenant appliquer l'algorithme de descente de gradient pour déterminer la trajectoire du robot. Le critère d'arrêt sera :

- soit que la distance entre la position q_{goal} et q_{robot} soit inférieure au pas δ_i
- soit un nombre maximum d'itérations.

Compléter le code afin de visualiser le chemin obtenu pour les cas définis précédemment.
Le code fourni ne réalise que le second critère. Ajouter le premier critère.
Conclure sur le comportement du robot selon les types de heuristiques utilisées.

3 – Extensions

Modifier le code pour traiter le cas de cible et/ou d'obstacles mobiles. Dans un premier temps, on considèrera des déplacements rectilignes (entre deux points). Ensuite, la cible pourra suivre une trajectoire circulaire.

Annexe : Best-First Planning Algorithm

La *recherche du chemin de moindre coût* consiste à échantillonner l'espace en une grille fine, puis à appliquer un algorithme de propagation, appelé Best-First Planning (BFP) :

```
Procedure BFP;
  begin
    instal q_init into T; [initially, T is the empty tree]
    INSERT (q_init, OPEN); mark q_init visited;
    [initially, all the configurations in C are marked unvisited]
    SUCCESS=false;
   while !empty(OPEN) or !SUCCESS do
      begin
         q<-FIRST(OPEN)
         for every node q' adjacent to q in C do
            if U(q')<M and q' is not visited then
              begin
                 add q' to T with a pointer toward q;
                 INSERT(q',OPEN); mark q' visited;
                  if q'=q_goal then SUCCESS <- true;
              end;
      end;
  if SUCCESS then
      return the constructed path by tracing the pointers in T from q_goal
      back to q_init
 else return failure;
end;
```

OPEN correspond à une pile,
INSERT(q,OPEN) insert la configuration q dans la pile OPEN.
q <- FIRST(OPEN) renvoie la configuration q de plus petit potentiel de la pile, et l'efface de cette pile.