Contrôleur pour un robot d'exploration

Amira Akloul, Hamza Belaoura, Maxime Kermarquer

Université de Versailles Saint-Quentin

24 janvier 2017

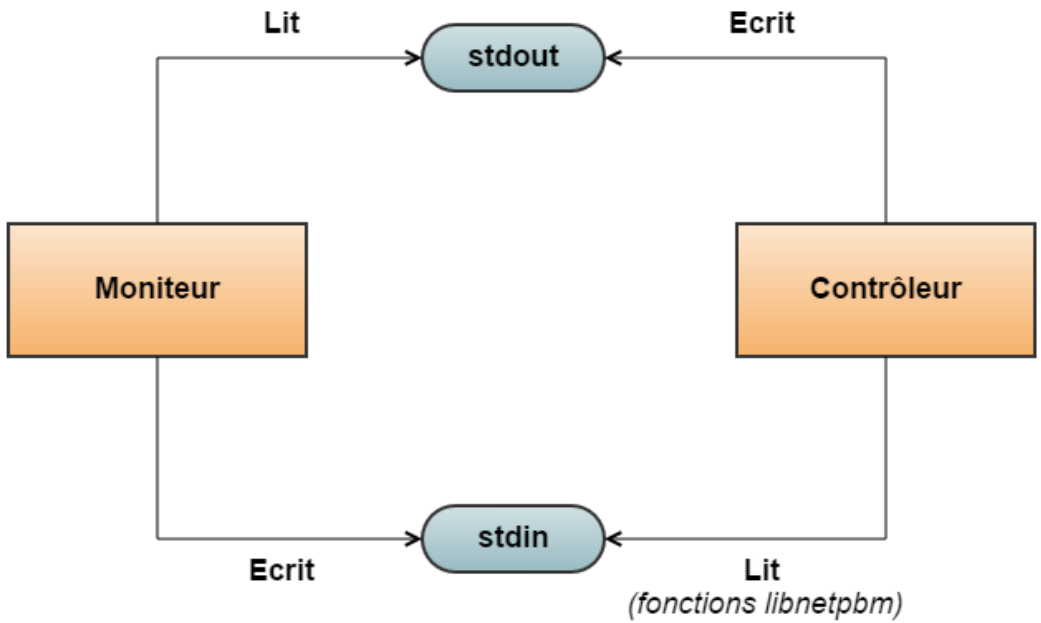
Sommaire

Introduction

- Objectifs du projet :
  - Développer un algorithme de navigation pour un Mars Rover.
  - Minimiser l'énergie dépensée par le Rover.
  - Minimiser l'empreinte mémoire du contrôleur.
- Outils à disposition :
  - Un simulateur : Simule les mouvements du Rover, visualisation de l'image de la caméra du robot.
  - Un moniteur : Permet d'interpréter les commandes données par notre contrôleur.
  - Un ensemble de cartes modélisant des terrains.
- Notre contrôleur est écrit en C.

Communication moniteur - contrôleur

- Utilisation des flux standards.



Outils du contrôleur

Afin d'évoluer sereinement dans son environnement et d'arriver à son objectif le Rover a besoin de certains outils :

- Calculer sa position (x,y).
- S'orienter vers l'objectif.
- Calculer sa distance à l'objectif.
- Calculer sa distance au prochain obstacle.
- Récupérer l'image de la caméra.

Outils du contrôleur

Calculer sa position (x,y)

A chaque mouvement FORWARD le Rover doit actualiser son ancienne position (x,y).

$\Rightarrow$ Utilisation de fonctions trigonométrique pour passer de coordonnées polaires à cartésiennes :

- $x' = x + r\cos\theta$
- $y' = y + r\sin\theta$

Où θ est l'orientation du robot, et r le nombre de mètres du FORWARD.

Outils du contrôleur

S'orienter vers l'objectif

Après avoir éviter un obstacle le Rover doit se ré-orienter vers l'objectif, en connaissant uniquement sa position et celle de l'objectif.

$\Rightarrow$ Utilisation de l'arc tangente :

- $\theta' = \arctan(\frac{y}{x})$

(Ici l'objectif est au point (0,0))

Outils du contrôleur

Calculer la distance au prochain obstacle

- Pour calculer la distance entre le Rover et son prochain obstacle, on doit utiliser l'image de la caméra.
- On va calculer l'ensemble des distances de chaque pixel de la ligne d'horizon.
- La distance Rover-Obstacle sera le minimum de cet ensemble.

Outils du contrôleur

Calculer la distance à l'objectif

Afin de connaître précisément la distance du FOWARD a donné au Rover pour le faire avancer vers l'objectif on doit calculer la distance Rover-Objectif.

$\Rightarrow$ Utilisation du théorème de Pythagore :

- $d(rover, objectif) = \sqrt{(x^2 + y^2)}$

Outils du contrôleur

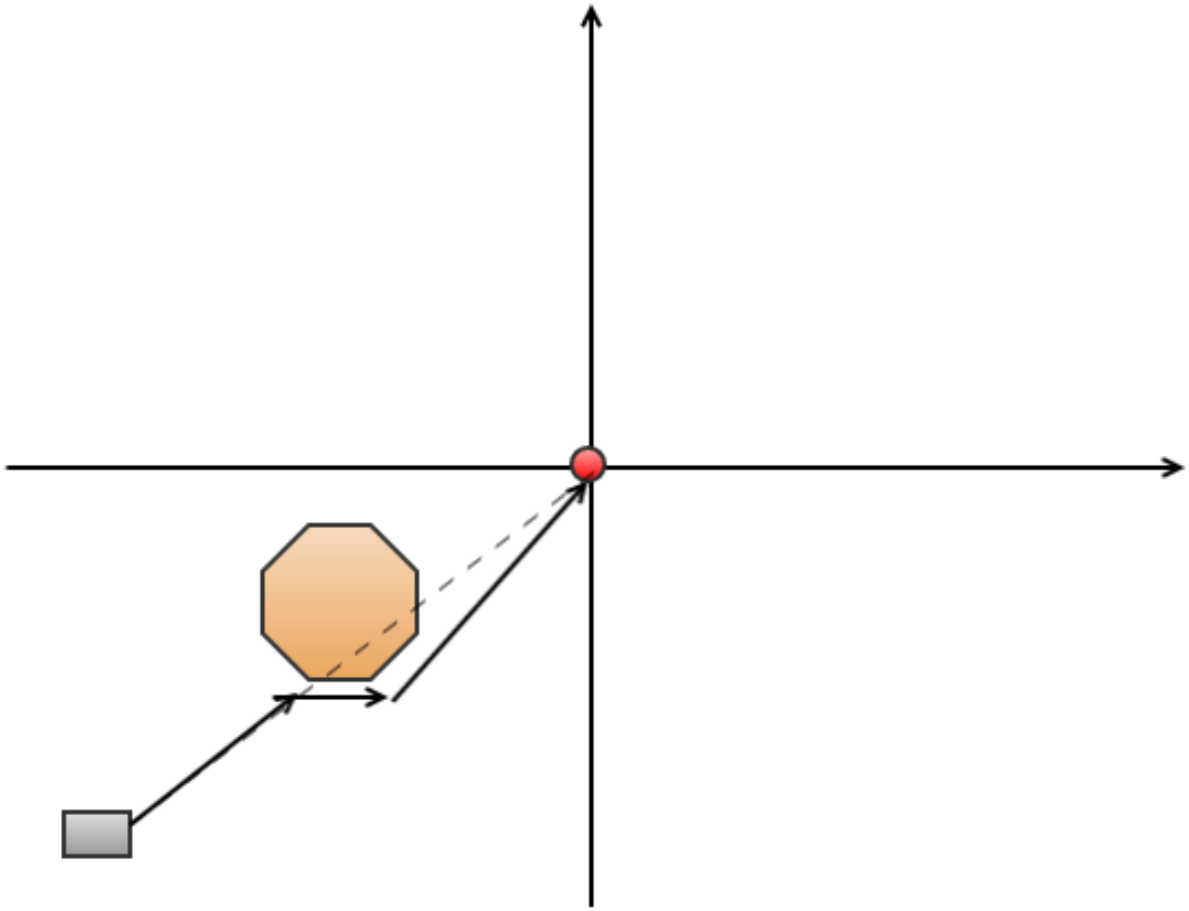
Récupérer l'image de la caméra

- Utilisation de la librairie *libnetpbm* qui lit et interprète le format PGM de manière précise et efficace.
- `#include <pgm.h>` (option de compilation : *-lnetpbm*)
- `void pgm_init (&argc, argv)`
- `gray ** pgm_readpgm (stdin, cols, rows, max)`
- `void pgm_freearray (gray **grays, int rows)`

Algorithme de navigation

while Rover n'est pas arrivé à l'objectif **do**
  CMD CAMERA
  CMD FORWARD min (d(Rover,Obstacle)-ϵ , d(Rover,OBJ))
  CALCUL_NOUVELLE_POSITION
  if Rover a avancé jusqu'à un obstacle **then**
    CMD TURN $-\frac{\pi}{4}$
    CMD CAMERA
    CMD FORWARD min (d(Rover,Obstacle)-ϵ , 5.0)
    CALCUL_NOUVELLE_POSITION
    CALCUL_ORIENTATION_OBJECTIF
    CMD TURN orientation_objectif
  end if
end while

Contournement d'obstacles



Conclusion

- Parties du projet fonctionnelles :
  - Fonctions de calcul de distances.
  - Calcul de position du Rover.
  - Calcul d'orientation du Rover vers l'objectif (à tester).
  - Récupération de l'image avec *libnetpbm* ...
- Choses à corriger, améliorer :
  - ... mais pour une image dans toute l'exécution du contrôleur !
  - Debuguer la fonction, ou développer une fonction qui parse la stdin pour récupérer l'image.
  - Optimiser la recherche de chemin avec un algorithme moins naïf.
  - Optimisation de l'empreinte mémoire (passage par adresse, récupérer uniquement la ligne d'horizon, ...)

Merci de votre attention.