

# Implémentation de l'algorithme de localisation de Monte Carlo en OCaml

Thomas Moulard

Master Parisien de Recherche en Informatique

03/02/2009

# Table des matières

- 1 Introduction
- 2 Algorithme de localisation de Monte Carlo (MCL)
- 3 Implémentation
- 4 Conclusion

# Table des matières

- 1 Introduction
- 2 Algorithme de localisation de Monte Carlo (MCL)
- 3 Implémentation
- 4 Conclusion

# Localisation et navigation en robotique

## Navigation

Action de conduire d'un point à un autre un robot en suivant un chemin donné.

## Localisation

Action de repérer sa position par rapport à un système de référence donné.

# Localisation et navigation en robotique

## Comment localiser un robot ?

- Comment définir un repère de référence ?
- Comment représenter les mouvements du robot dans ce repère ?

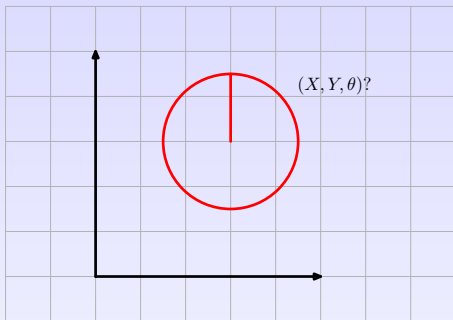


FIG.: Problème de la localisation d'un robot

# Définition du modèle du mouvement

## Modèle du mouvement

Vecteur  $(X, Y, \theta)^a$  représentant la translation des trois coordonnées durant une unité de temps.

<sup>a</sup>cas d'un robot à roues se déplaçant en 2D.

Modèle du mouvement :

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} 0 \\ +4 \\ 0 \end{pmatrix}$$

Mouvement réel :

$$\begin{pmatrix} x + \eta_x \\ y + \eta_y \\ \theta + \eta_\theta \end{pmatrix}$$

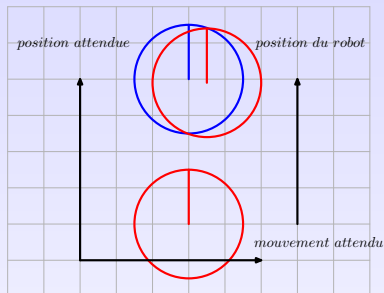


FIG.: Modèle de mouvement

# Algorithme de Monte Carlo

## Problème rencontré

- Peut déterminer la vitesse linéaire/angularaire réelle d'un robot (odométrie et encodeurs sur les roues).
- ...cependant ce n'est pas suffisant en pratique (dérapage...).

## Algorithme de localisation de Monte Carlo

Donne une distribution de positions  $(X, Y, \theta)$  associés une probabilité (filtre à particule). Le mode de la distribution donne l'emplacement réel du robot.

# Algorithme de Monte Carlo

## Avantages

- Simple.
- Efficace
- Gère les distributions multi-modales (ambiguïté).

## Limites

- Peut échouer.
- Doit borner les erreurs.

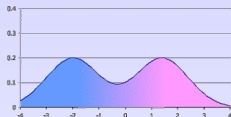


FIG.: Distribution bimodale<sup>a</sup>

---

<sup>a</sup>Image tirée de Wikipedia.

# Table des matières

- 1 Introduction
- 2 Algorithme de localisation de Monte Carlo (MCL)**
- 3 Implémentation
- 4 Conclusion

# Etape 0 : initialisation

## Initialisation des particules

Soit  $N$  le nombre de particules utilisées. Les particules sont placées aléatoirement autour de la position initiale avec une probabilité uniforme  $1/N$  (distribution gaussienne).

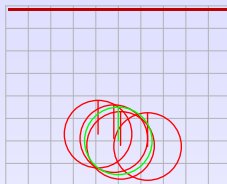


FIG.: Initialisation des particules

# Etape 1 : mise à jour de la position

## Mise à jour des particules

On utilise le modèle du mouvement pour mettre à jour chaque particule.

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \\ \theta_{n+1} \end{pmatrix} = \begin{pmatrix} x_n + dx + \eta_x \\ y_n + dy + \eta_y \\ \theta_n + d\theta + \eta_\theta \end{pmatrix}$$

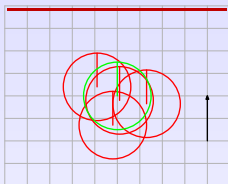


FIG.: Mise à jour des particules

## Etape 2 : mise à jour des scores

### Mise à jour des scores

La probabilité associée à chaque particule est inversement proportionnelle à l'erreur de mesure entre les valeurs attendues et réelles de chaque capteur.

- Pour chaque capteur, pour chaque particule :  

$$score = score * \exp^{-err^2 / (2 * noise^2)}$$
- ... puis on normalise en divisant par la somme des scores  

$$(\sum_{i=0}^n w_i = 1).$$

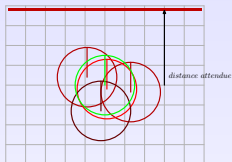


FIG.: Mise à jour des scores

# Etape 3 : rééchantillonnage

## Rééchantillonnage (si nécessaire)

On calcule  $\sum_{i=1}^n (w_i - \bar{w})^2$  (ESS). Si  $ESS < seuil$  alors on rééchantillonne (en dupliquant les particules les plus "saines").

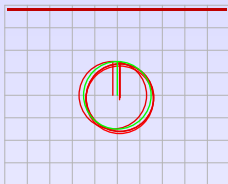


FIG.: Rééchantillonnage

# Table des matières

- 1 Introduction
- 2 Algorithme de localisation de Monte Carlo (MCL)
- 3 Implémentation**
- 4 Conclusion

# MCLML

## Démo

- Sans mise à jour des scores.
- Avec un capteur.
- Avec deux capteurs.

# Table des matières

- 1 Introduction
- 2 Algorithme de localisation de Monte Carlo (MCL)
- 3 Implémentation
- 4 Conclusion**

# Conclusion




## Bilan

- Algorithme simple, rapide, généralisable facilement. . .
- . . . mais algorithme probabiliste, peut échouer dans des cas complexes,
- . . . ou ne pas pouvoir départager deux positions similaires.
- Simulation limitée par la connaissance “parfaite” des obstacles.

## Perspectives

- Gérer les “kidnapping” (ajouter régulièrement de l’aléa aux particules).
- Implémentation *anytime*.
- *SLAM* : ajout d’une méthode pour cartographier l’environnement en parallèle (Wavefront).

# Bibliographie

-  Carpenter, J., Clifford, P., and Fernhead, P. (1997).  
An improved particle filter for non-linear problems.
-  Thrun, S., Burgard, W., and Fox, D. (2005).  
*Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*.  
The MIT Press.
-  Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001).  
Robust monte carlo localization for mobile robots.  
*Artificial Intelligence*, 128(1-2) :99–141.