



UNIVERSITÉ  
LAVAL

# GLO-4001/7021 INTRODUCTION À LA ROBOTIQUE MOBILE

## SLAM

**Rappel : -présentation orales vendredi**

# Introduction au SLAM

# Vers le SLAM...

$z$  : mesure

$x$  : pose

$u$  : commandes

$m$  : carte

- Modèle de capteur :  $p(z | x)$
- Localisation avec repères :  $p(x | z, m)$
- Filtrage KF/EKF/Part. :  $p(x_t | z_{1:t}, u_{1:t}, m)$

- SLAM :  $p(x_t, m | z_{1:t}, u_{1:t})$  *filtrage*

On veut estimer la pose et la carte

- Full-SLAM :  $p(x_{1:t}, \hat{m} | z_{1:t}, u_{1:t})$  *lissage/smoothing*

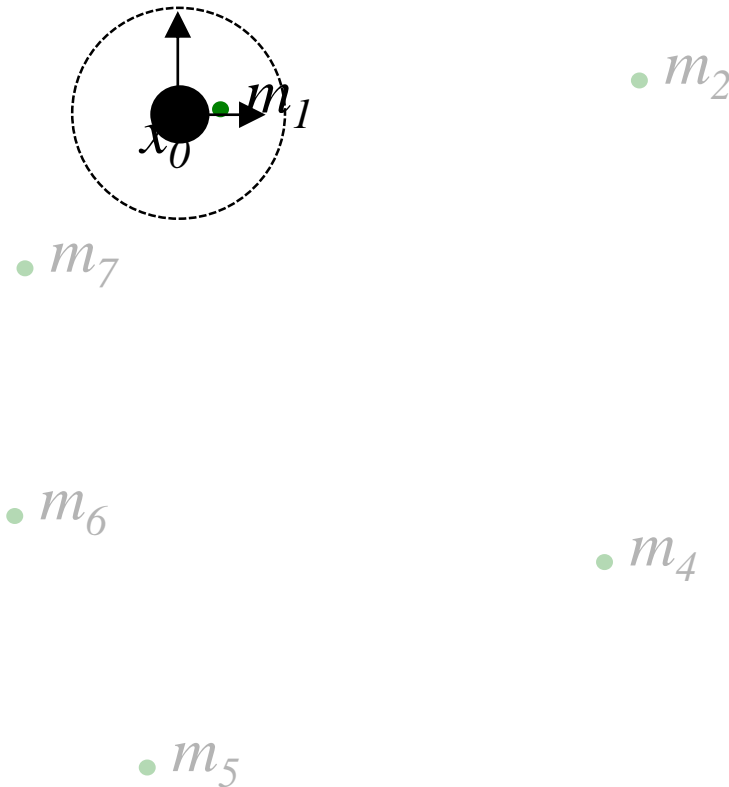
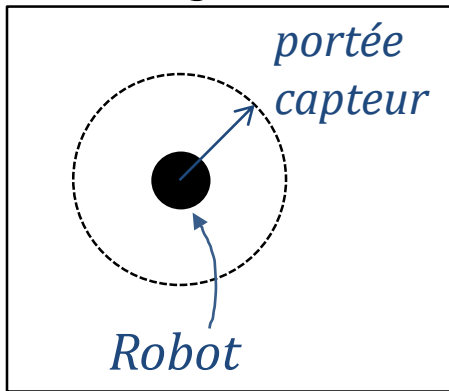


trajectoire complète

# SLAM : scénario

- Un robot explore un monde inconnu...

Légende



point de repère  
 $m_i = [m_{i,x}, m_{i,y}, s_i]^T$   
signature

carte  $m = [m_1 \dots m_n]^T$

# SLAM : Simultaneous Localization and Mapping

---

- 2 Buts :
  - créer une carte  $m$  de l'environnement
$$m = [m_1, m_2, m_3, \dots, m_M]^T \text{ (} M \text{ points de repères)}$$
$$m_i = [m_{i,x}, m_{i,y}]^T \text{ en 2D}$$
  - maintenir un estimé  $X$  de la pose d'un robot par rapport à cette carte  $m$
- Les mesures  $z$  donnent l'information entre les repères dans  $m$  et la pose du robot  $X$ .
- Donne l'incertitude des poses (*sur les points de repère et le robot*)
- Considéré comme un des problèmes les plus importants en robotique mobile

# Défis...

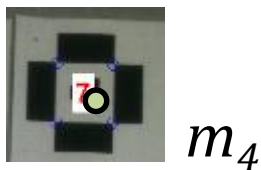
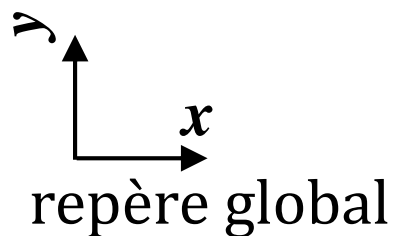
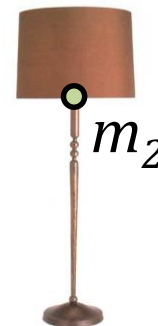
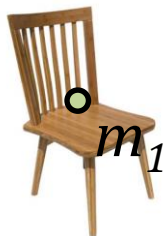
---

- Existes des méthodes robustes pour des environnements statiques, structurés et de taille limité
- Défis pour des environnements :
  - non-structurés (extérieur)
  - dynamiques (voitures, piétons)
  - changeants (météo, illumination, saison, construction)
  - grande échelle ( $10 \text{ km}^2+$ )

# Carte $m$

Points de repères visuels : position  $x, y$  et signature  $s$

↓  
SIFT,  
Marqueur fiduciaire

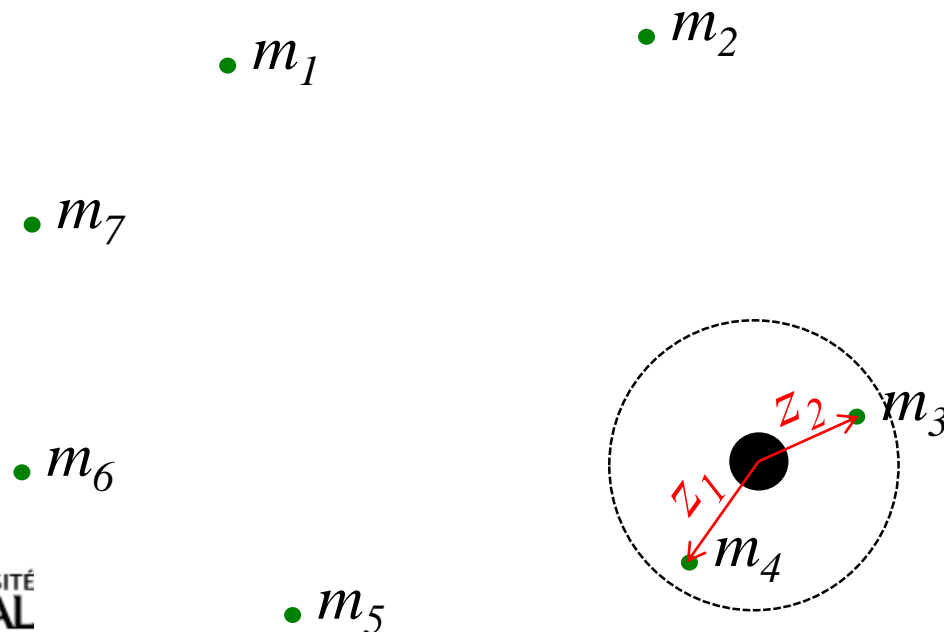


# SLAM : Simultaneous Localization and Mapping

- **Information connue :**

- commandes envoyées (odométrie)  $U_{1:T} = [\vec{u}(1), \vec{u}(2), \dots, \vec{u}(T)]^T$
- mesures des capteurs  $Z_{1:T} = [\vec{z}(1), \vec{z}(2), \dots, \vec{z}(T)]^T$  (e.g. distance+direction)
- appariement mesures-repères  $C_{1:T} = [\vec{c}(1), \vec{c}(2), \dots, \vec{c}(T)]^T$ 
  - associer une mesure du vecteur  $\vec{z}(t)$  au point de repère  $m_i$   
(data association problem)

transpose  
↓



À l'instant t, on mesure

$$\vec{z}(t) = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \begin{array}{l} \rightarrow m_4 \\ \rightarrow m_3 \end{array}$$

Le vecteur  $\vec{c}$  associe  $z_1$  à  $m_4$ , et  $z_2$  à  $m_3$  : **DATA ASSOCIATION!**



# SLAM : Simultaneous Localization and Mapping

---

- **Information connue :**

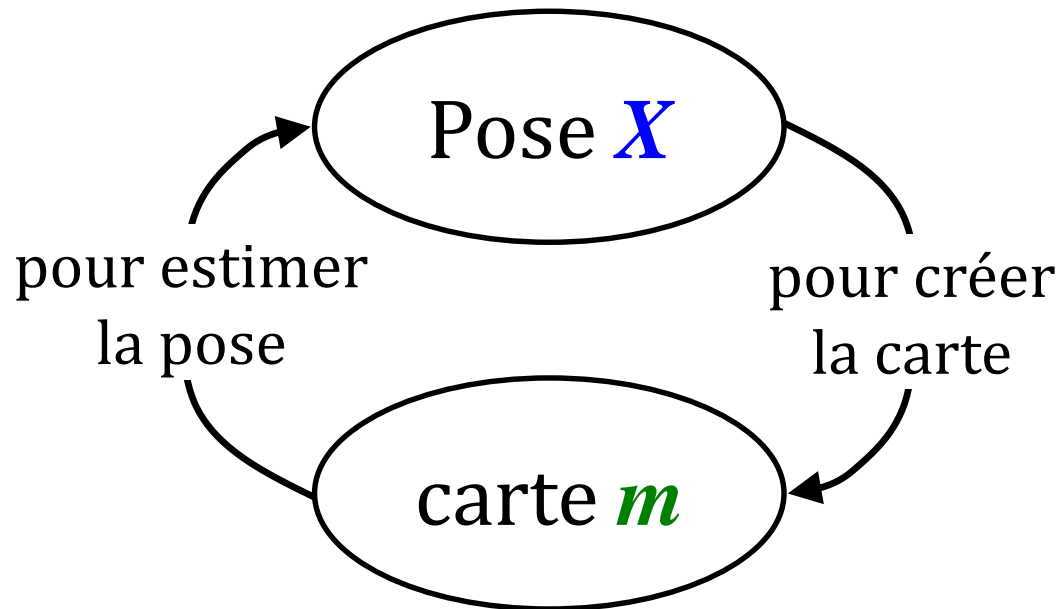
- commandes envoyées (odométrie)  $U_{1:T} = [\vec{u}(1), \vec{u}(2), \dots, \vec{u}(T)]^T$
- mesures des capteurs  $Z_{1:T} = [\vec{z}(1), \vec{z}(2), \dots, \vec{z}(T)]^T$
- appariement mesures-repères  $C_{1:T} = [\vec{c}(1), \vec{c}(2), \dots, \vec{c}(T)]^T$ 
  - associer une mesure du vecteur  $\vec{z}(t)$  au point de repère  $m_i$
- modèle du déplacement du robot  $f_X(\mathbf{X}, \mathbf{u})$
- modèle des capteurs  $h_z(\mathbf{X}, m)$
- bruits Gaussiens présents dans le système

- **Inconnu**

- $X_{0:T} = [\vec{x}(0), \dots, \vec{x}(T)]^T$  : trajectoire du robot
- $m$  : carte du monde (ensemble points de repère  $m_i$ )

# SLAM : problème circulaire

Paradoxe de l'œuf ou la poule...



On devra donc résoudre ces deux problèmes  
(estimation de  $X$  et de  $m$ ) en même temps

# SLAM : types de problèmes

---

- Full SLAM (lissage/smoothing)
  - Trajectoire complète et carte

$$p(X_{1:t}, m | Z_{1:t}, U_{1:t})$$

*Souvent des algorithmes de type batch*

- SLAM en ligne (filtrage)
  - quelle est ma position actuelle  $X(t)$ ?

$$p(X_t, m | Z_{1:t}, U_{1:t})$$

*Algorithmes en ligne*

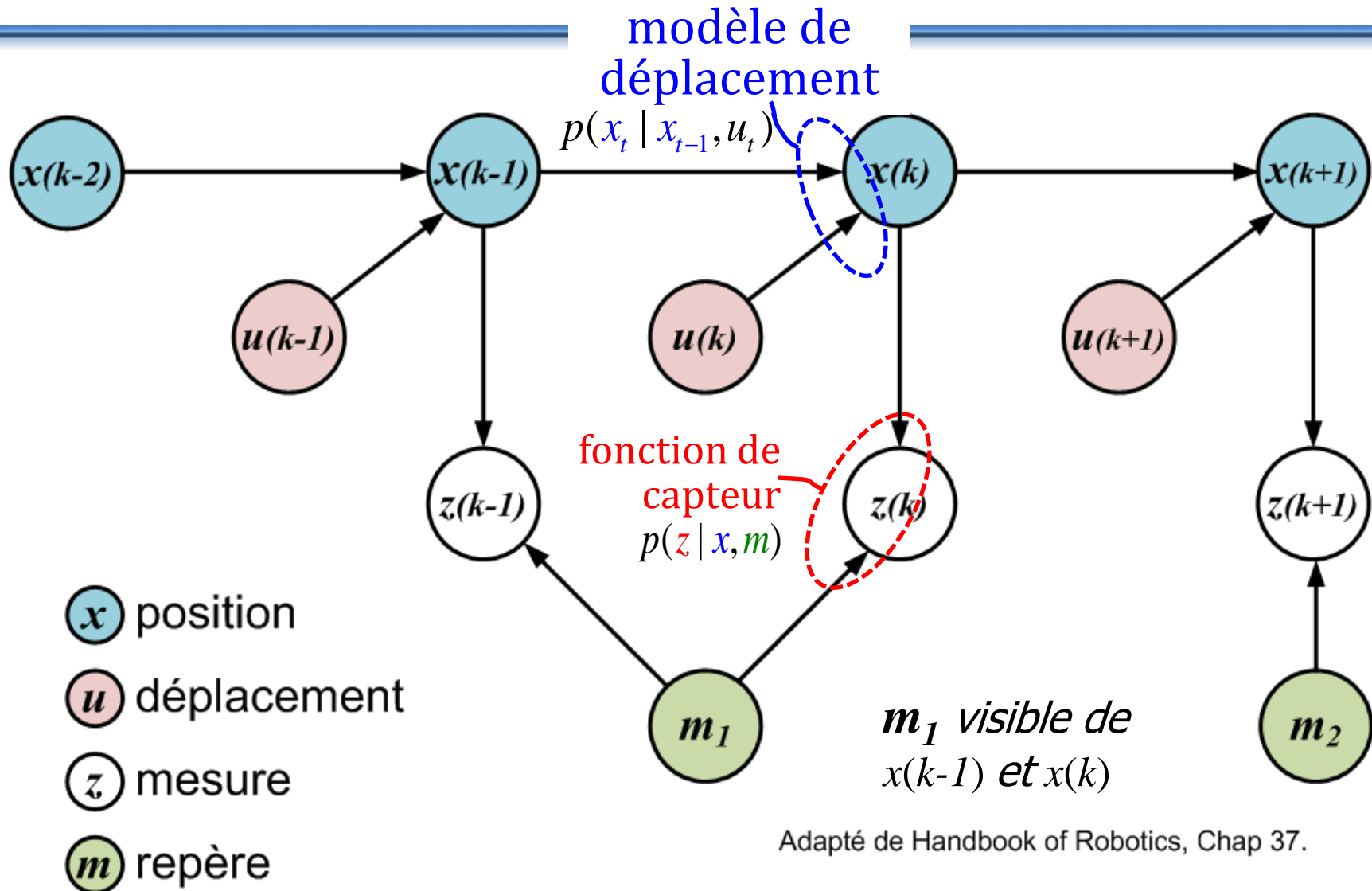
*(filtrage : on ne corrige pas le passé)*

# Hypothèses sur les repères

- Les repères  $m_i$  sont des points :
  - 2D : coin de mur, cadre de porte
  - 3D : repère visuel (coin de cadre, objets, etc.)
- Facilement identifiables et distinguables
  - problème d'association mesure entre  $z_j$  et repère  $m_i$



# Graphe des dépendances



Adapté de Handbook of Robotics, Chap 37.

**Important : repère  $m_i$  n'est pas nécessairement visible de partout**

# Taxonomie

---

- **Volumétrique (dense) vs. Features**  
permet de recréer des images photo-réalistes (erreur de reprojection)
- **Topologique vs. Métrique**  
Relation entre les lieux      On peut calculer des distances partout
- **Correspondances connues vs. Inconnues**
- **Statique vs. Dynamique**

# Taxonomie

---

- Actif vs. Passif
  - SLAM contrôle l'exploration
  - SLAM n'envoie aucune commande au robot
- Single-robot vs. Multi-robot
  - Autre robot visible ou non
  - Bande passante, délais, *packet loss*, etc.

# 3 méthodes populaires pour SLAM

---

- Filtre Kalman Étendu (EKF)
- Filtres à particules
- Méthodes graphiques : Graph SLAM



**EKF-SLAM**

# Filtrage Bayésien : rappel

---

- Filtrage récursif en deux phases

- **Prédiction**

$$bel_{pred}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

- **Mise-à-jour**

$$bel(x_t) = \eta p(z_t | x_t) bel_{pred}(x_t)$$

# Filtre de Kalman étendu (EFK) : rappel

- (1)  $\hat{x}(k+1|k) = f_x(\hat{x}(k), u(k))$  déplacement du robot
- (2)  $\Phi = \left. \frac{df_x}{dx} \right|_{\hat{x}(k+1)}$        $\Gamma = \left. \frac{df_x}{du} \right|_{\hat{x}(k+1)}$  calcul des Jacobiennes, pour propager erreur
- (3)  $P(k+1|k) = \Phi P(k) \Phi^T + \Gamma C_v \Gamma^T$  augmentation covariance P, après déplacement (le bruit ici est sur la commande)
- (4)  $\hat{z}(k+1|k) = h_z(\hat{x}(k+1|k))$  prédire ce que je devrais mesurer
- (5)  $r(k+1) = z(k+1) - \hat{z}(k+1|k)$  compare avec mesure
- (6)  $\Lambda^T = \left. \frac{dh_z}{dx} \right|_{\hat{x}(k+1)}$  calcul du Jacobien pour la mesure
- (7)  $K(k+1) = P(k+1|k) \Lambda^T \{ \Lambda P(k+1|k) \Lambda^T + C_w(k+1) \}^{-1}$  mélange optimal
- (8)  $\hat{x}(k+1) = \hat{x}(k+1|k) + K(k+1)r(k+1)$  combine estimé pose + mesure
- (9)  $P(k+1) = (I - K(k+1)\Lambda) P(k+1|k)$  ajuste covariance P, après gain info

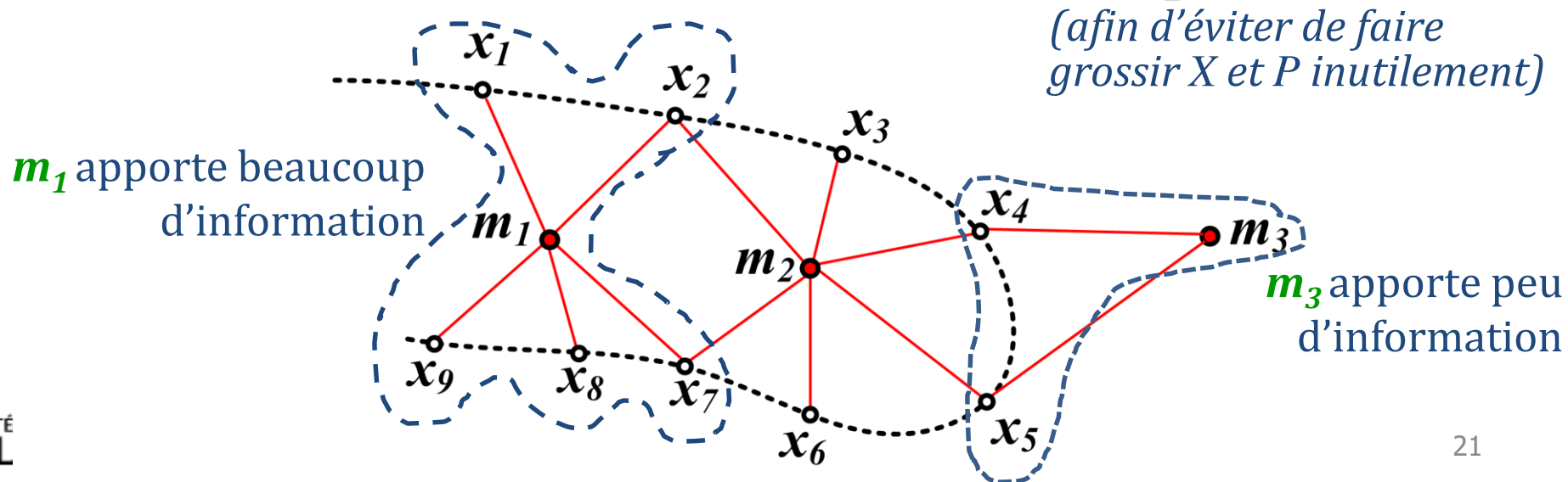
# SLAM + Kalman Étendu (EKF) = EKF-SLAM

- On ajoute les repères  $m_i$  dans le vecteur d'état :

$$X = \begin{bmatrix} x & y & \theta & \boxed{m_{1,x} & m_{1,y}} & \boxed{m_{2,x} & m_{2,y}} \end{bmatrix}^T$$

$m_1$   $m_2$

- On ajoute les  $m_i$  quand ils sont découverts.
- De préférence, on ajoute les  $m_i$  qui sont visibles à de nombreux endroits, i.e. visibles fréquemment.



# SLAM + Kalman Étendu (EKF)

- La matrice de covariance sera beaucoup plus grande

$$X = \begin{bmatrix} x & y & \theta & m_{1,x} & m_{1,y} & m_{2,x} & m_{2,y} \end{bmatrix}$$



pour  $M=2$  points de repère :  $P = [7 \times 7]$

- Taille de  $P$  est  $O(n^2)$ ,  $n = |X| =$  taille état =  $3+2M$  (en 2D)
- $P$  est non-creuse, contient l'information mutuelle entre les repères  $m_i$  et poses du robot
- Temps de calcul total pour une carte :  $O(n^2)$ 
  - limite en pratique l'application de la méthode

# Algorithme EKF-SLAM (2D)

---

- Ajout d'un nouveau repère

$$\begin{bmatrix} m_{j,x} \\ m_{j,y} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \text{pose relative } x \\ \text{pose relative } y \end{bmatrix}$$

nouveau  
repère

pose  
estimée  
du  
robot

mesure relative

# Algorithme EKF-SLAM (2D)

## *Prédiction*

Prédiction de  $x$ ,  $y$  et  $\theta$  (robot) dans  $X$ , en fonction de  $u$

Augmentation de la covariance de la pose du robot dans  $P$

## *Mise-à-jour*

Prise des mesures  $z_t^j$  des  $j$  repères visibles

Pour chaque repère visible  $j$  (*donc autant de mise-à-jour que de repères visibles*)

Apparier mesure  $j$  avec repère  $m_j$ , selon la signature :  $\vec{c}$

Si  $j$  est un nouveau repère, l'ajouter\* dans  $X$

Estimer la mesure  $\hat{z}_t^i$

Calculer la covariance  $Q$  associée à la mesure  $z_t^j$

Calculer le gain  $K_j$  selon  $P$ ,  $Q$  et jacobienne de mesure  $\Lambda$

Ajuster l'état  $X$  et  $P$ , selon  $K_j$  et l'innovation  $z_t^j - \hat{z}_t^i$

\*ou le mettre dans une liste provisoire

# Algorithme EKF-SLAM

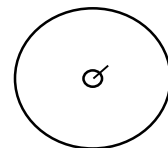
---

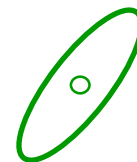
- Plus complexe car
  - les vecteurs/matrices augmentent au fil du temps
  - le nombre de mesures varie dans le temps
  - il faut apparier les mesures  $z_t^j$  aux bons repères  $m_i$
  - le nombre de mise-à-jour dépend du nombre de repères visibles




# Démo EKF sur matlab

- Bruits sur déplacements :  $\sigma_V = 0.05$  m/s,  $\sigma_\omega = 0.1^\circ$ /s
- Bruits sur mesures des repères en polaire :  $\sigma_r = 0.2$  m,  $\sigma_\phi = 10^\circ$

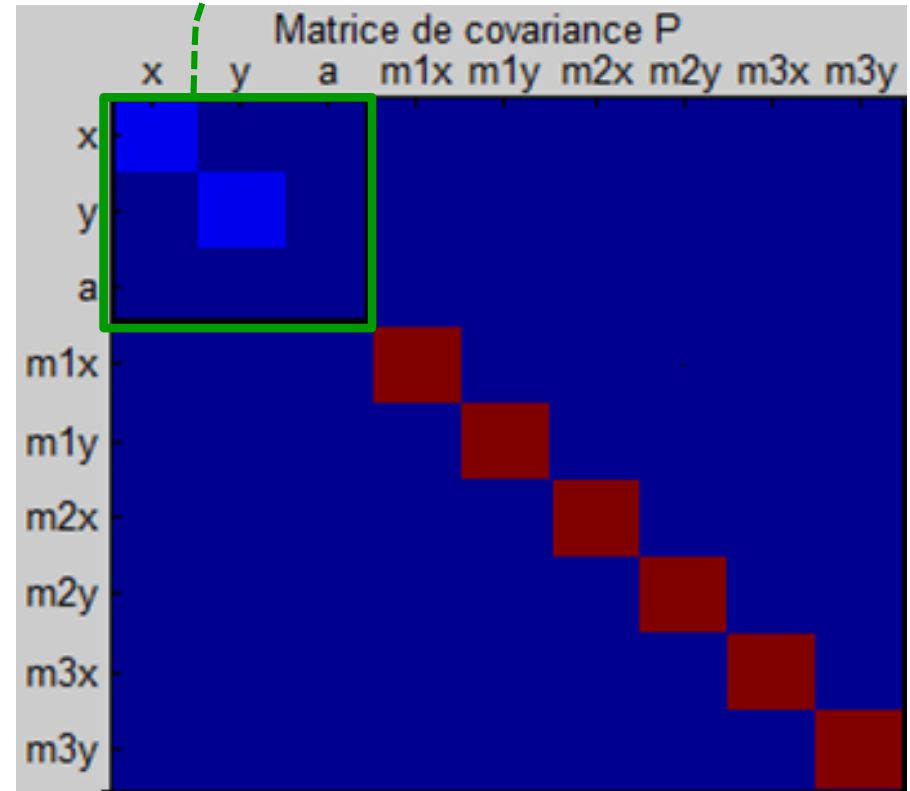
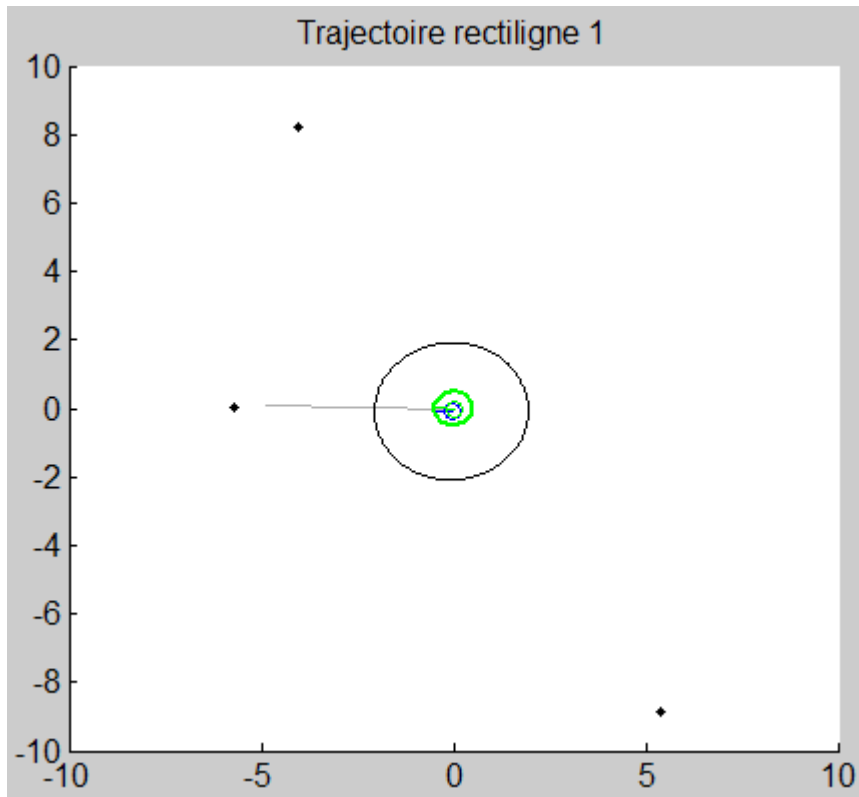
 *pose réelle et portée du capteur*

 *covariance de l'estimé de pose  $X$*

 *covariance de l'estimé d'un repère  $m_i$*

# Exemple EKF-SLAM

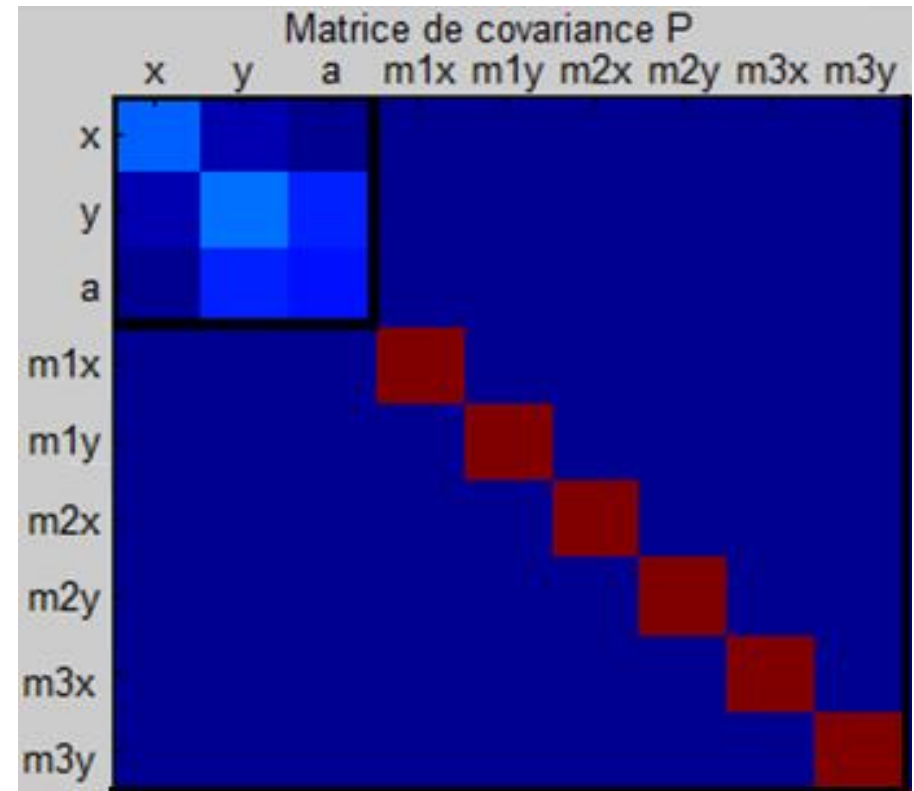
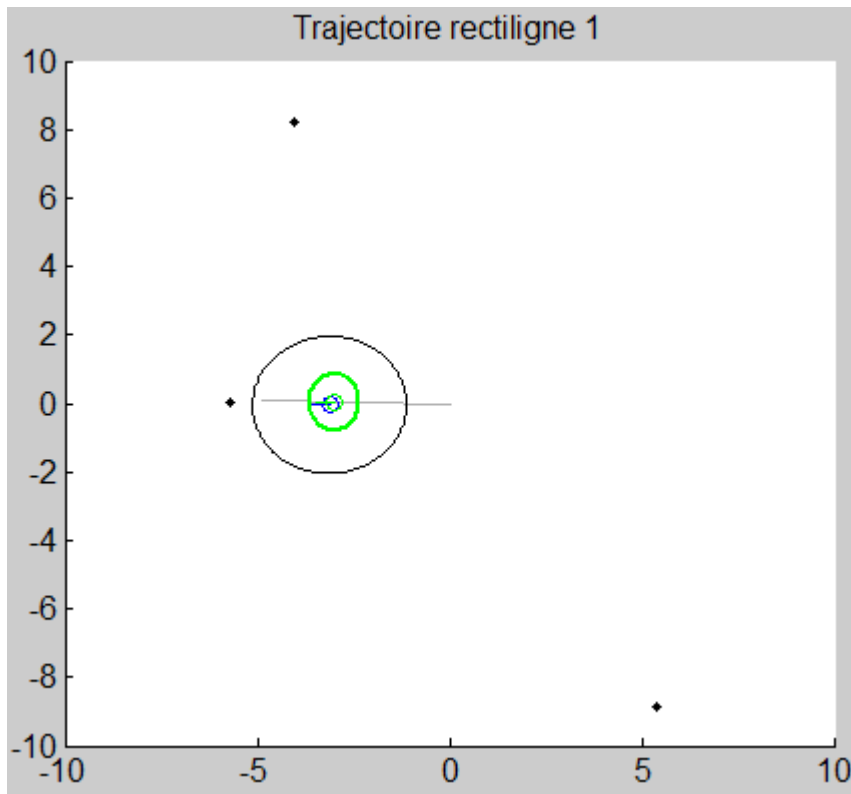
- Au début



Note : ici on initialise P en assumant qu'il y a exactement 3 repères dans le monde.

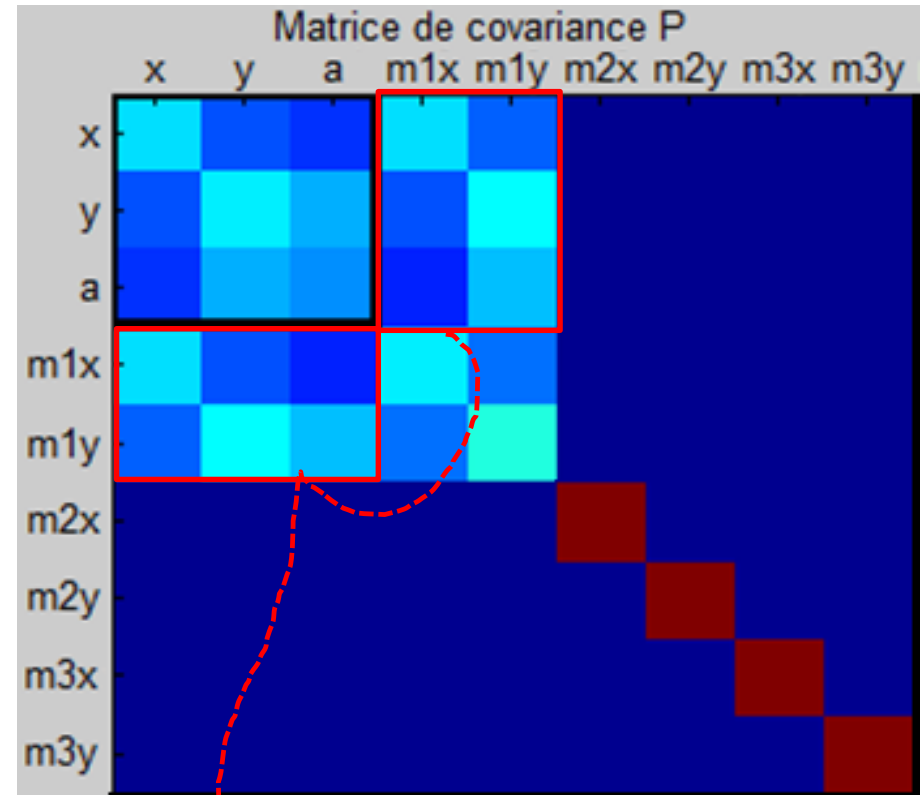
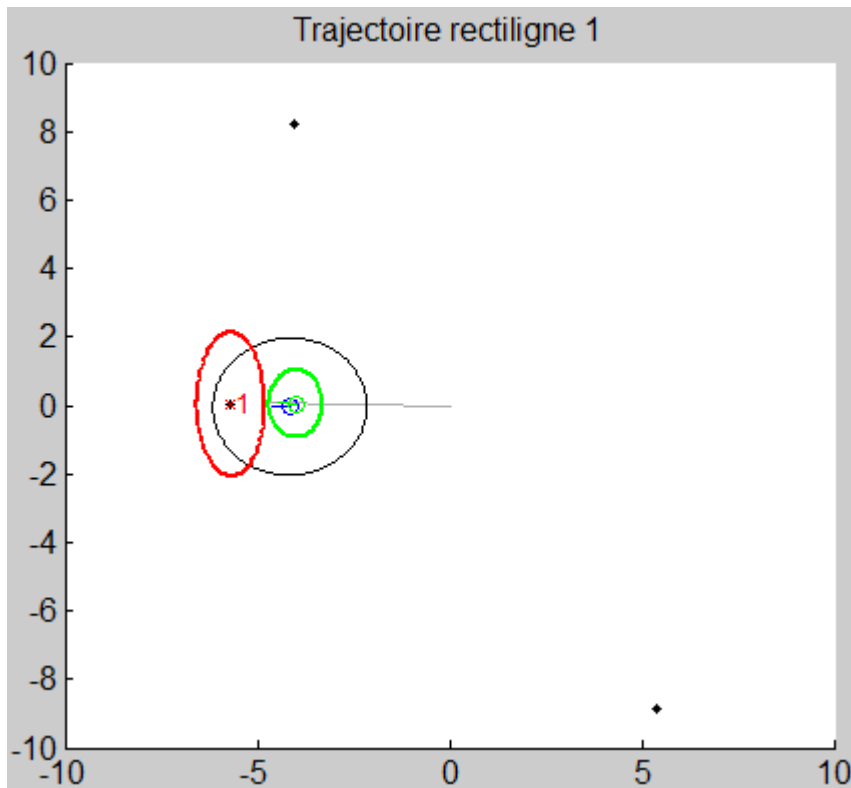
# Exemple EKF-SLAM

- En route : incertitude croît



# Exemple EKF-SLAM

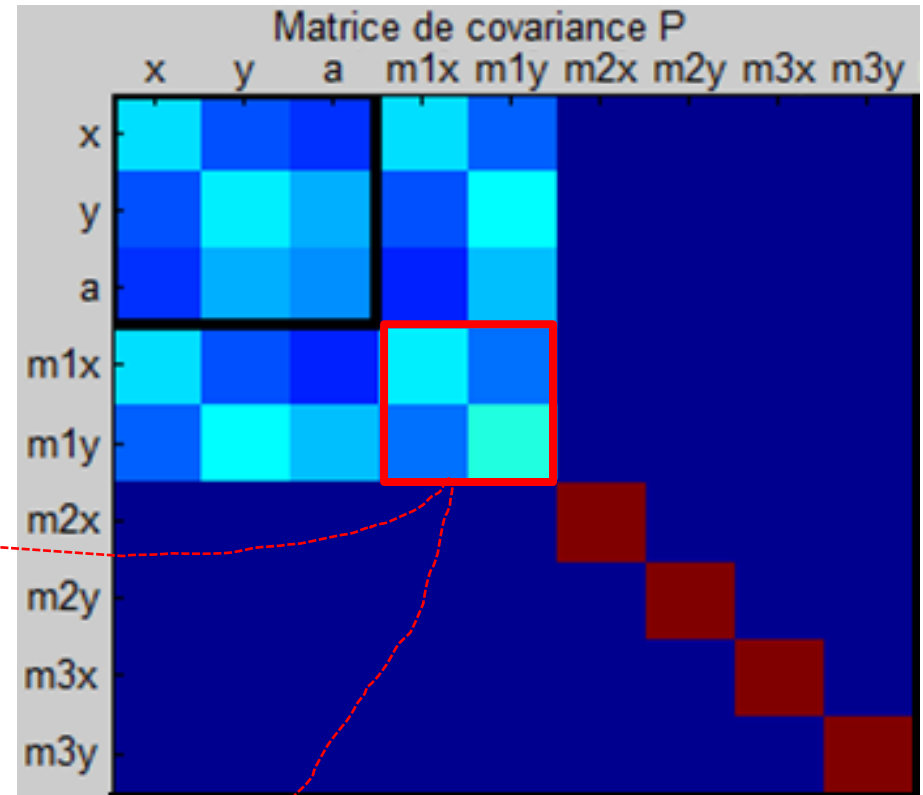
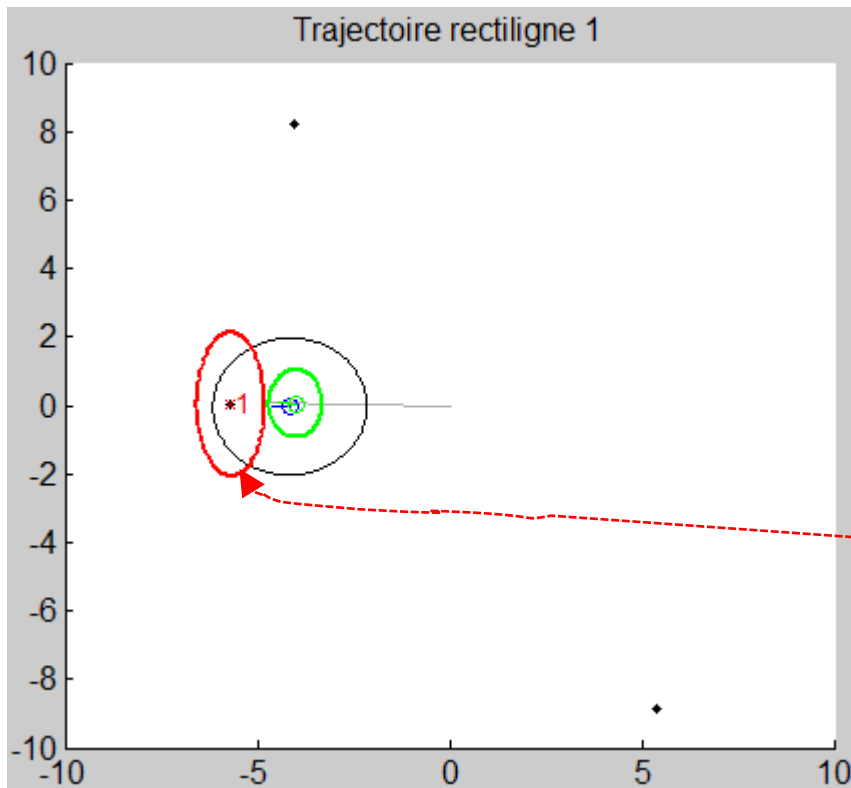
- Rencontre du premier repère : ajout dans  $X$  et  $P$



Indique corrélation entre pose du robot et repère  $m_1$

# Exemple EKF-SLAM

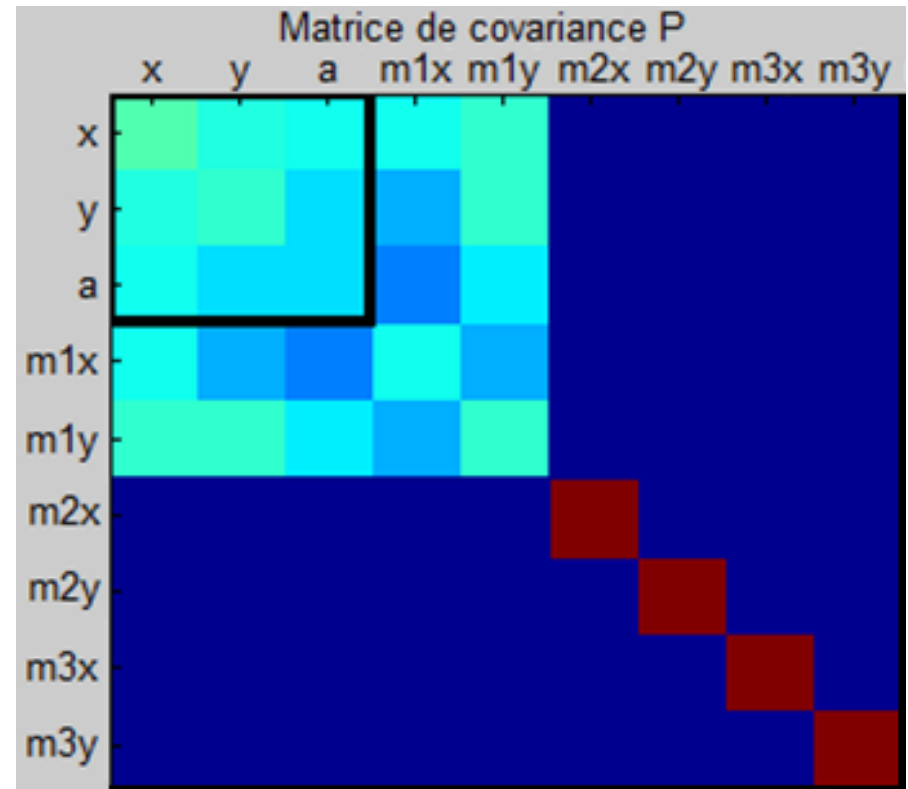
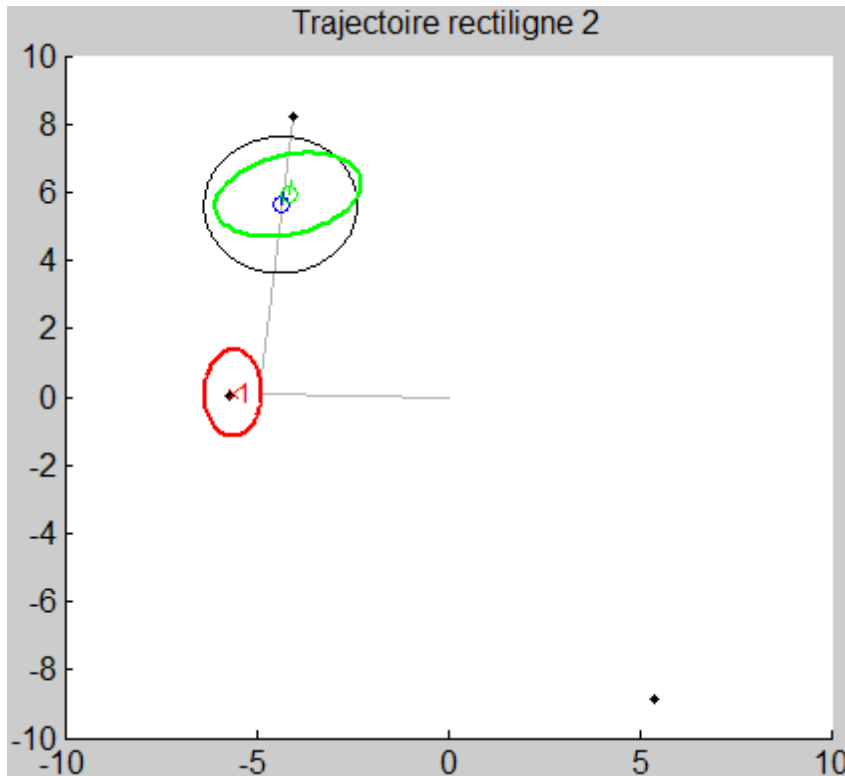
- Rencontre du premier repère : ajout dans  $X$  et  $P$



Indique l'incertitude sur la pose du repère  $m_1$

# Exemple EKF-SLAM

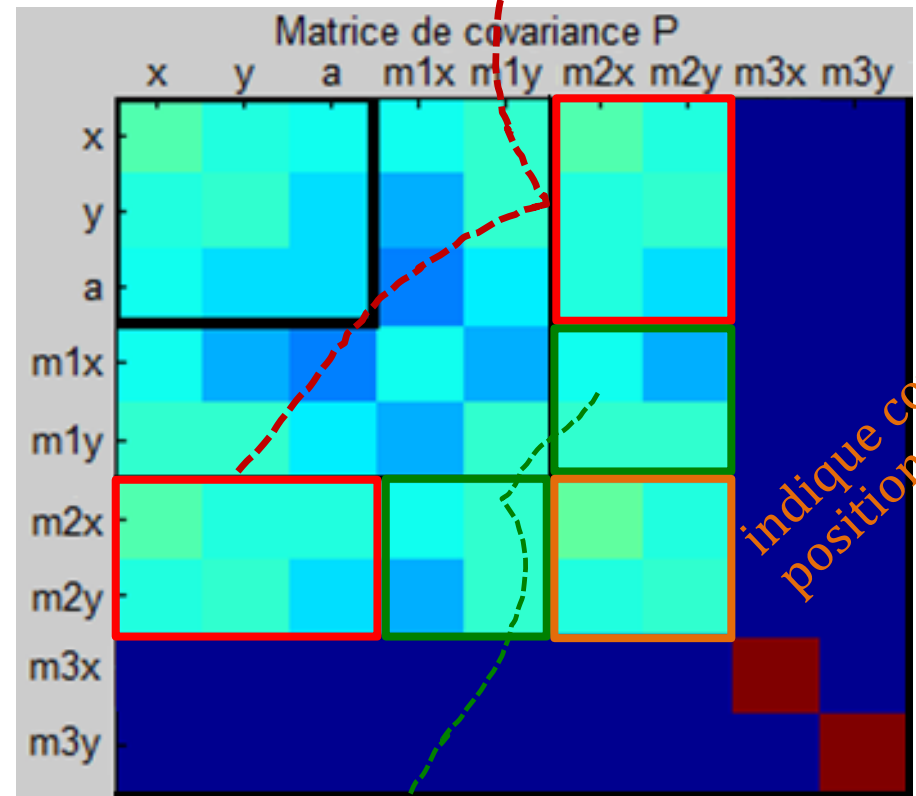
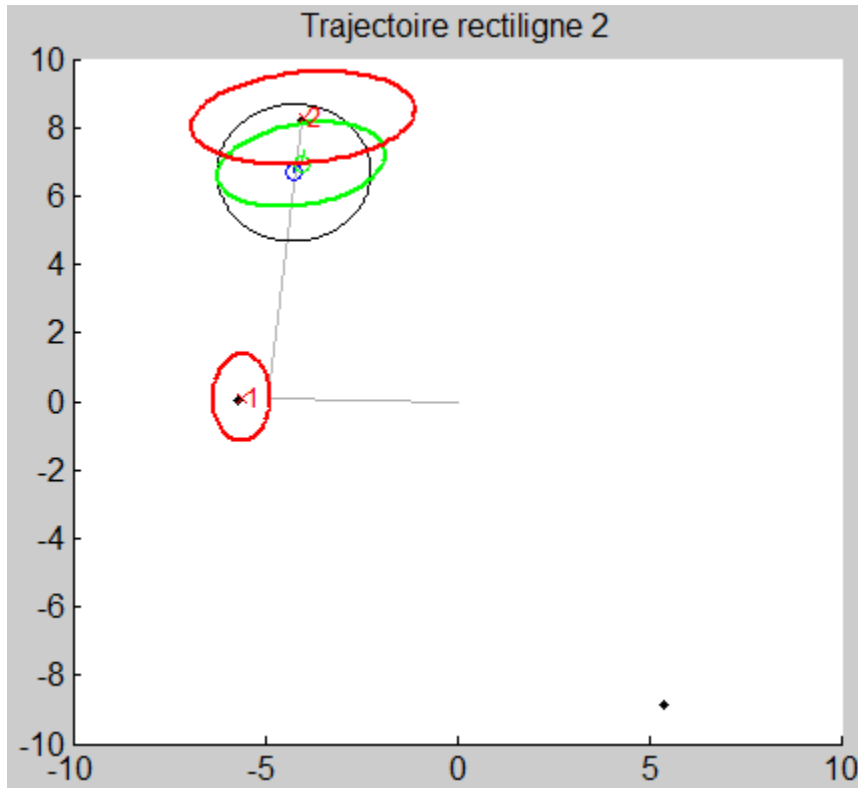
- Juste avant deuxième repère. Incertitude croît



# Exemple EKF-SLAM

- Ajout du deuxième repère

Indique lien entre pose du robot et repère  $m_2$



indique cov. sur position de  $m_2$

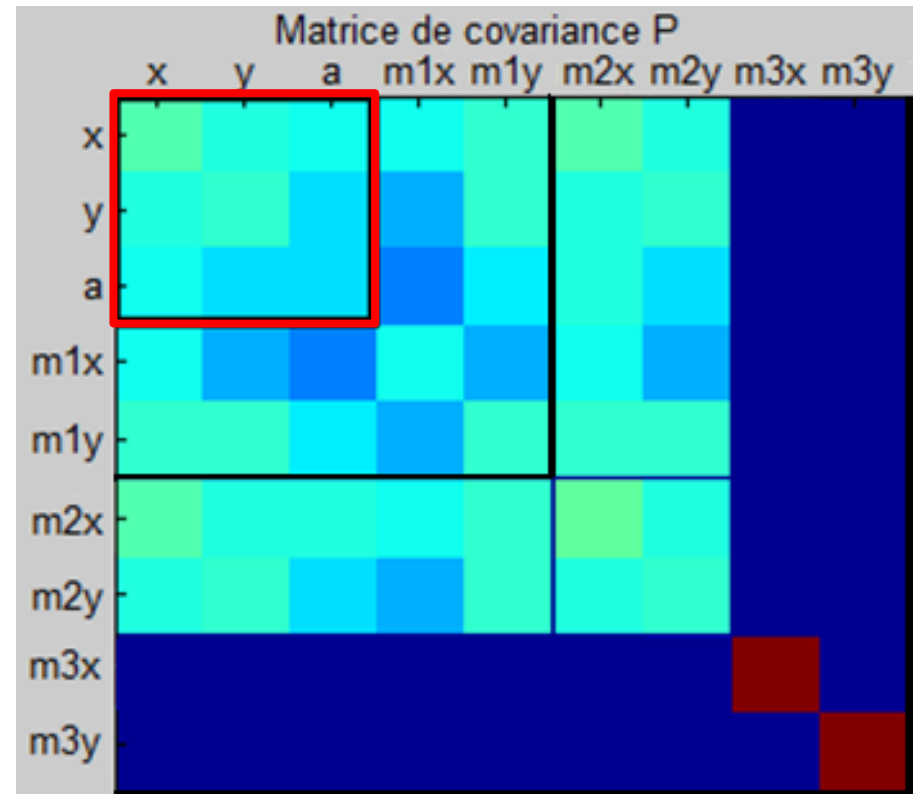
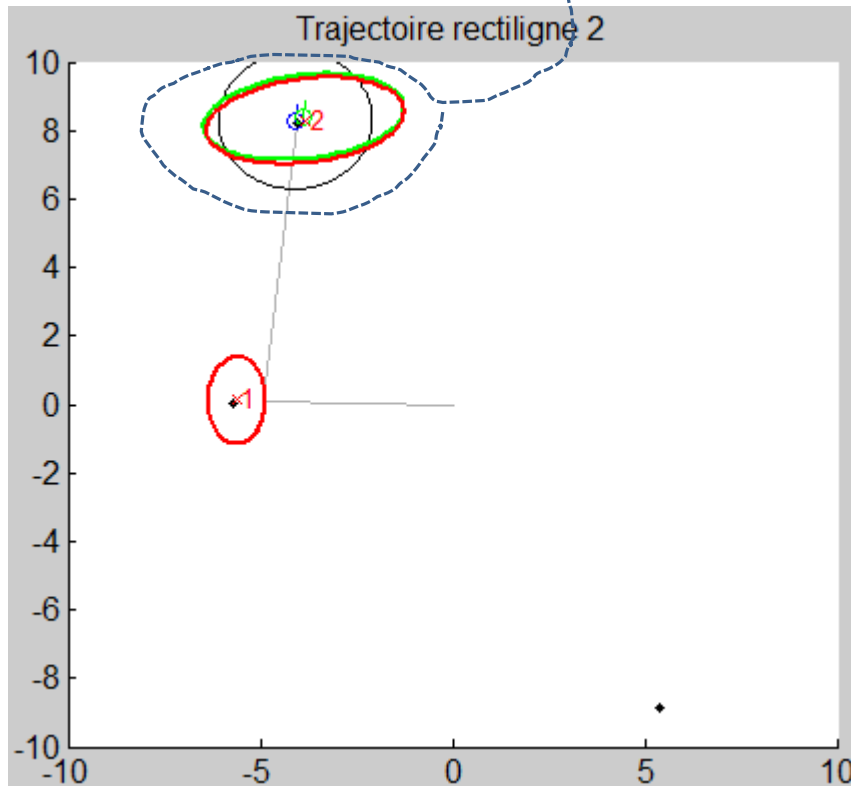
(Concrètement, cela veut dire que si je modifie  $m_1$ ,  $m_2$  sera automatiquement ajusté)



Indique corrélation entre repère  $m_1$  et repère  $m_2$

# Exemple EKF-SLAM

- On voit bien que l'incertitude du repère provient de l'incertitude de la pose du robot



*note : capteur précis à courte portée*

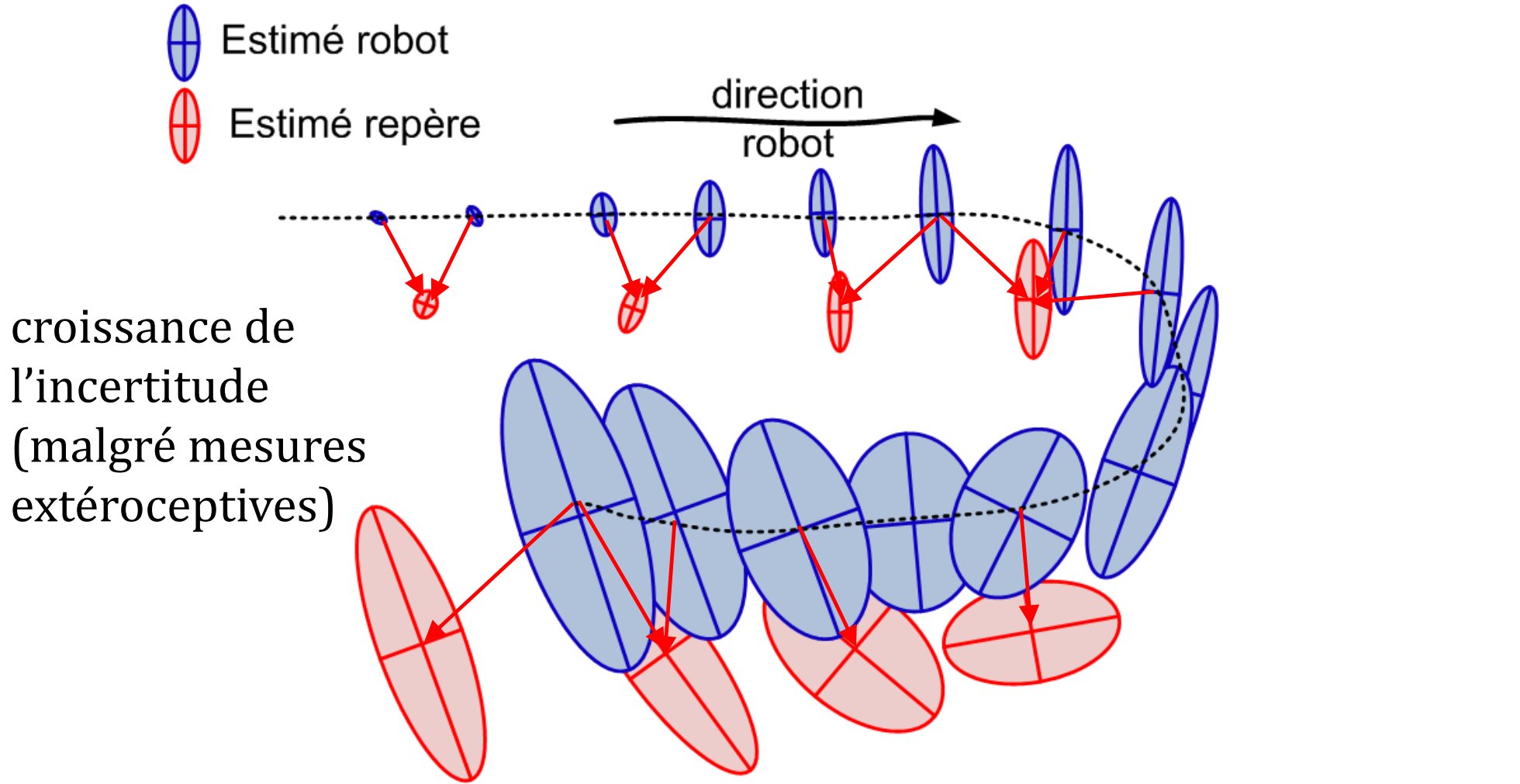


# Exemple EKF-SLAM

---

- Script matlab

# SLAM : fermeture de boucle



*Inspiré de Michael Montemerlo, Stanford U.*

# SLAM : fermeture de boucle

 Estimé robot

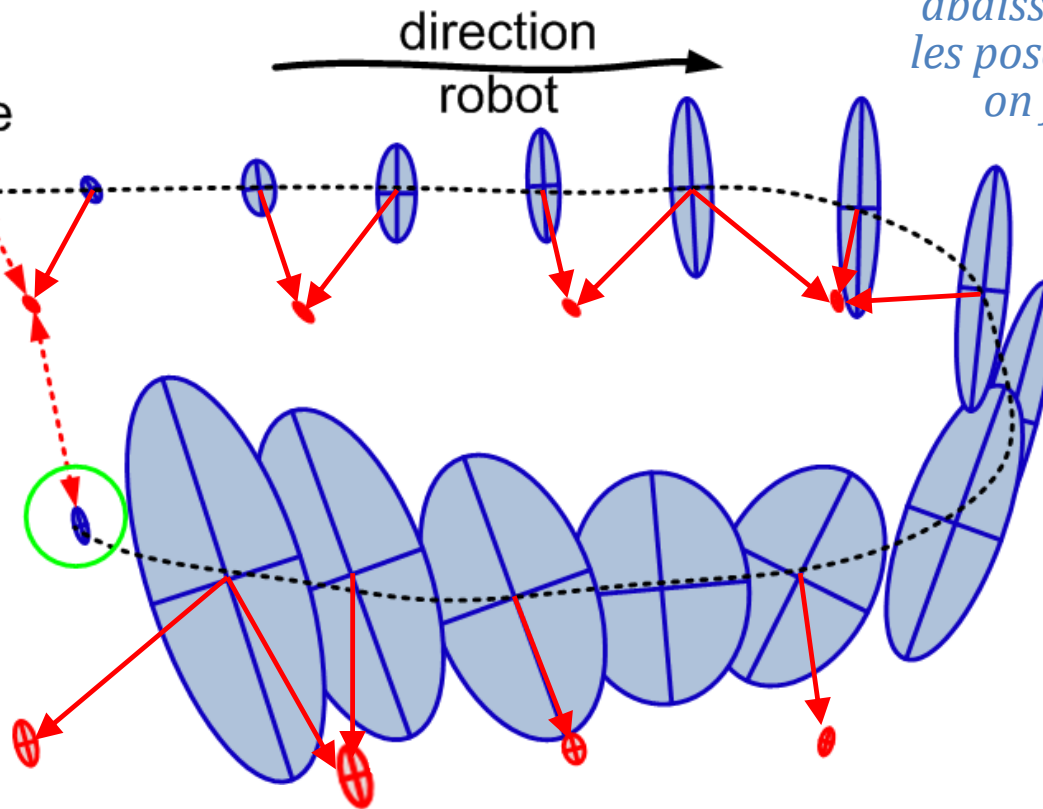
 Estimé repère

*Note : ici on n'a pas abaissé l'incertitude sur les poses antérieures, car on fait du filtrage*

**boucler la boucle...**

**baisse soudaine d'incertitude sur la pose du robot + les position des repères**

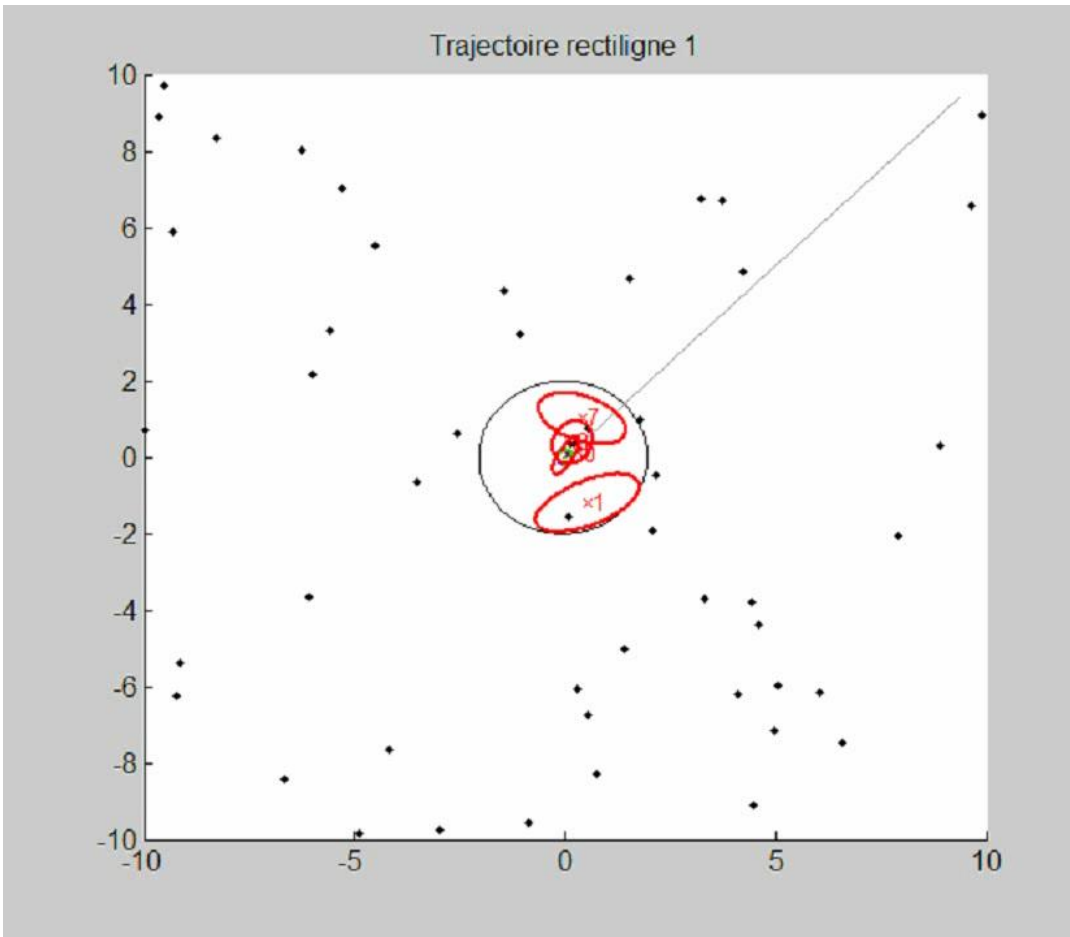
*(propagé via les covariances entre les repères, dans la matrice P)*

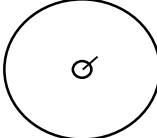


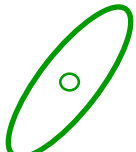
**La détection des boucles est fondamentale à la convergence du SLAM**


# EKF-SLAM : baisse de l'incertitude sur $m$

- Bruits sur déplacements :  $\sigma_V = 0.05$  m/s,  $\sigma_\omega = 0.1^\circ$ /s
- Bruits sur mesures des repères en polaire :  $\sigma_r = 0.2$  m,  $\sigma_\phi = 10^\circ$



 *pose réelle et portée du capteur*

 *covariance de l'estimé de pose  $X$*

 *covariance de l'estimé d'un repère  $m_i$*

À la limite, l'incertitude finale sur la carte dépend de l'incertitude du début

# EKF-SLAM : résumé

---

- On ajoute la carte  $m$  dans l'état  $X$
- On propage le robot comme dans EKF
- Mesures extéroceptives de la mise-à-jour viennent diminuer l'incertitude
  - de la position du robot
  - de la carte  $m$
- Important de revisiter les points de repères du début (ils ont le minimum d'incertitude)  
*(la trajectoire du robot aura un impact très important sur la précision de la carte!)*
- Temps de calcul  $O(M^2)$  à chaque étape  
*(imaginez si vous avez  $M=1,000$  points de repères...)*

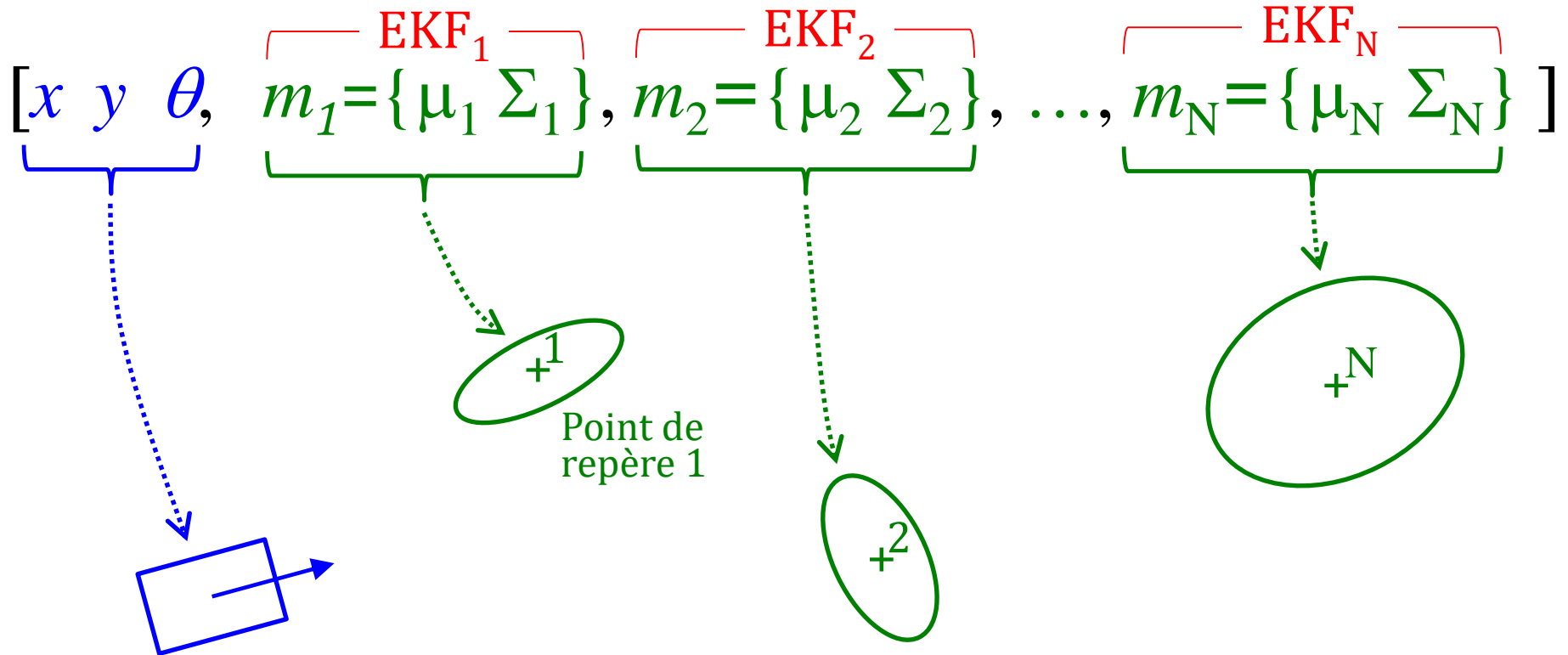
FastSLAM

# SLAM avec filtre à particules?

---

- Ici, une particule contient la pose du robot + carte  $m$
- Nombre de dimensions très élevé :  $N = 50$ , on aura 103 dimensions... ☹️
- Solution : Rao-Blackwellization

# FastSLAM 1.0 : Particule



Où la carte est  $m = \{m_1, m_2, \dots, m_N\}$



# FastSLAM 1.0 : Algorithme

---

```
for each particle
  p.X = propagate(p.X,U,motion_noise) (simule un déplacement, avec bruit)
  for each mesure  $z_i$ 
    p.w = p.w * p( $z_i$  | p.X) (est-ce que ma carte et la pose p.X sont vraisemblables, selon les mesures?)
  end
end
NormalizeParticleWeights()
ResampleParticles() (élimine les mauvaises cartes/poses)
for each particle
  for each mesure  $z_i$ 
    EKF_Update(p.landmark $_i$ ,  $z_i$ ) (mise-à-jour de  $\mu_i$  et  $\Sigma_i$ )
  end
end
```

*(pas de prédiction EKF)*

*Hypothèse : on connaît parfaitement l'association landmark-mesure*

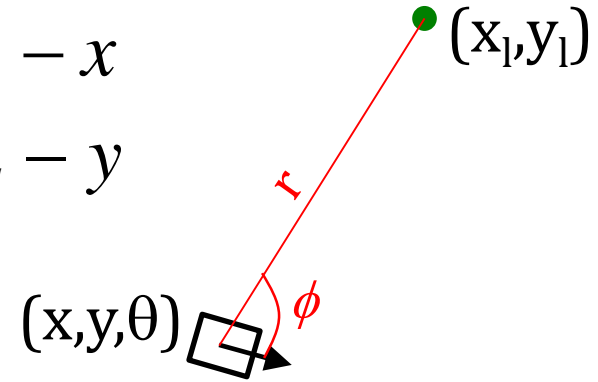
# Update EKF

$$z = \begin{bmatrix} r \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{\Delta x^2 + \Delta y^2} \\ \text{atan2}(\Delta y, \Delta x) - \theta \end{bmatrix}$$

$$\Delta x = x_l - x$$

$$\Delta y = y_l - y$$

$$\Lambda = \begin{bmatrix} \frac{\partial r}{\partial x_l} & \frac{\partial r}{\partial y_l} \\ \frac{\partial \phi}{\partial x_l} & \frac{\partial \phi}{\partial y_l} \end{bmatrix} = \begin{bmatrix} \frac{\Delta x}{r} & \frac{\Delta y}{r} \\ -\frac{\Delta y}{r^2} & \frac{\Delta x}{r^2} \end{bmatrix}$$



$$P = \Sigma_l \quad \hat{x}(k+1/k) = \mu_l$$

$$\hat{z}(k+1|k) = h_z(\hat{x}(k+1|k))$$

$$r(k+1) = z(k+1) - \hat{z}(k+1|k)$$

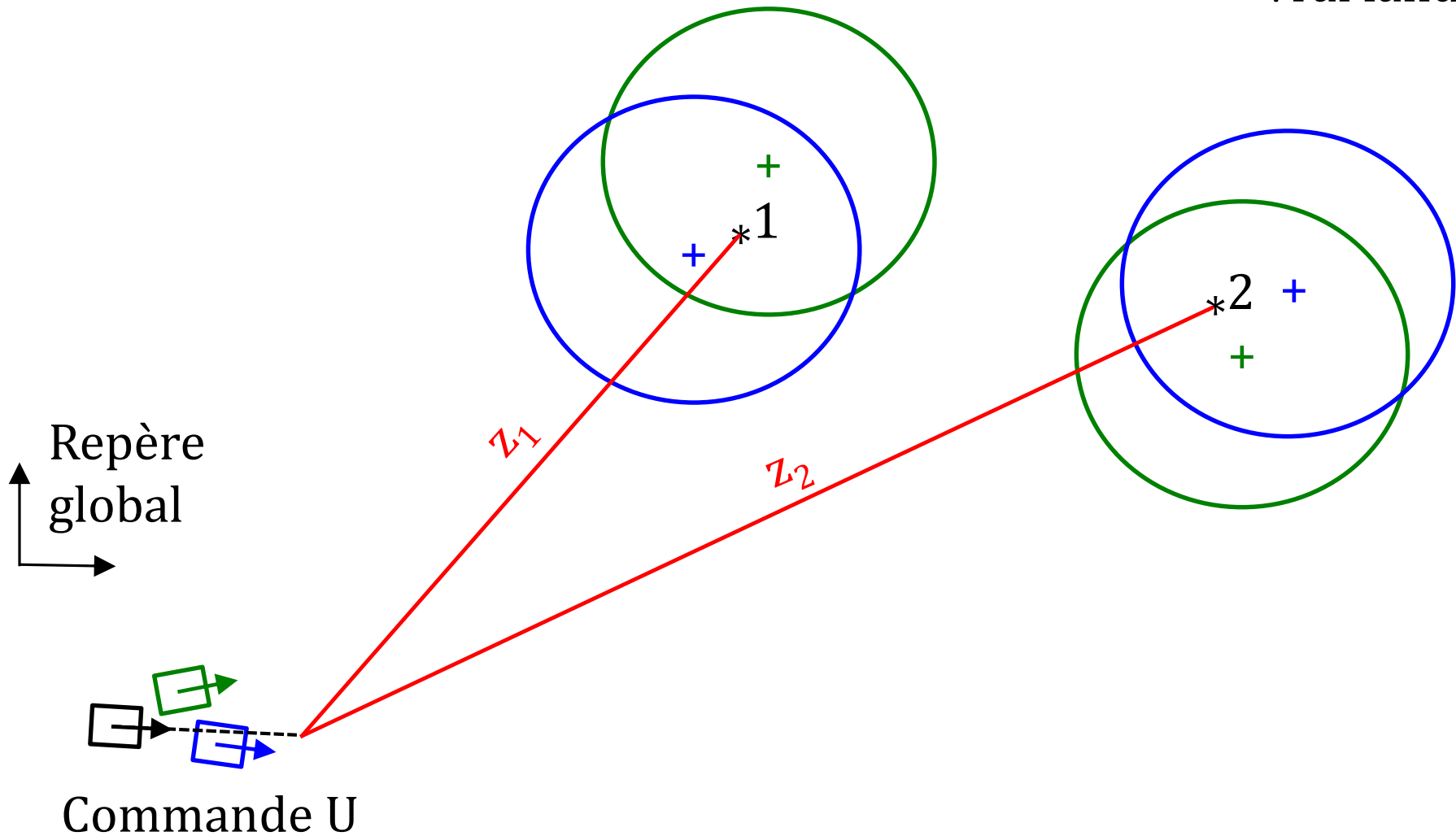
$$K(k+1) = P(k+1|k)\Lambda^T \{\Lambda P(k+1|k)\Lambda^T + C_w(k+1)\}^{-1}$$

$$\hat{x}(k+1) = \hat{x}(k+1|k) + K(k+1)r(k+1)$$

$$P(k+1) = (I - K(k+1)\Lambda)P(k+1|k)$$

# FastSLAM 1.0 : 2 particules

\* Vrai landmark



Commande U

# Fichier matlab

---

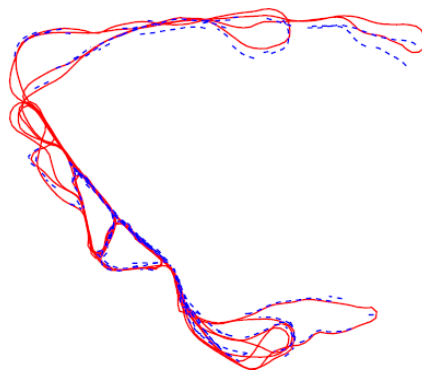
- Run it!

# Résultats FastSLAM 1.0

odométrie seulement



FASTSlam

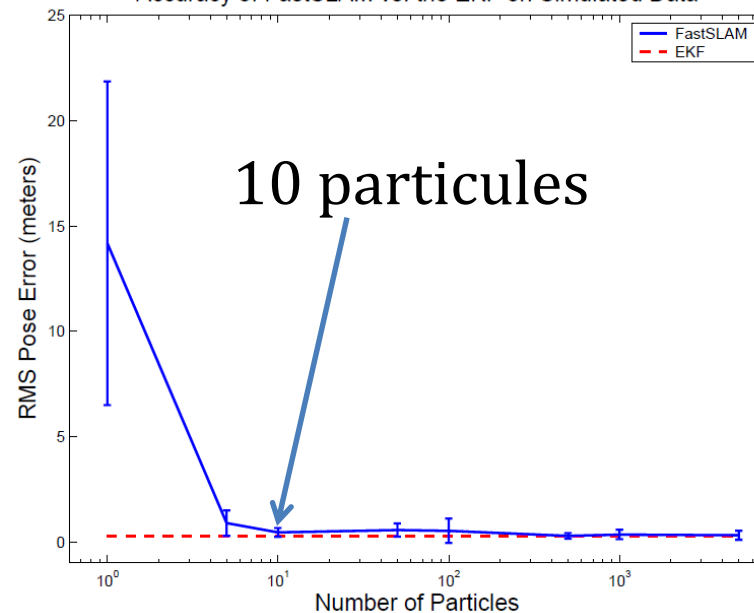


30 minutes, 4 km



Télémètre laser

Accuracy of FastSLAM vs. the EKF on Simulated Data



points de repère : tronc

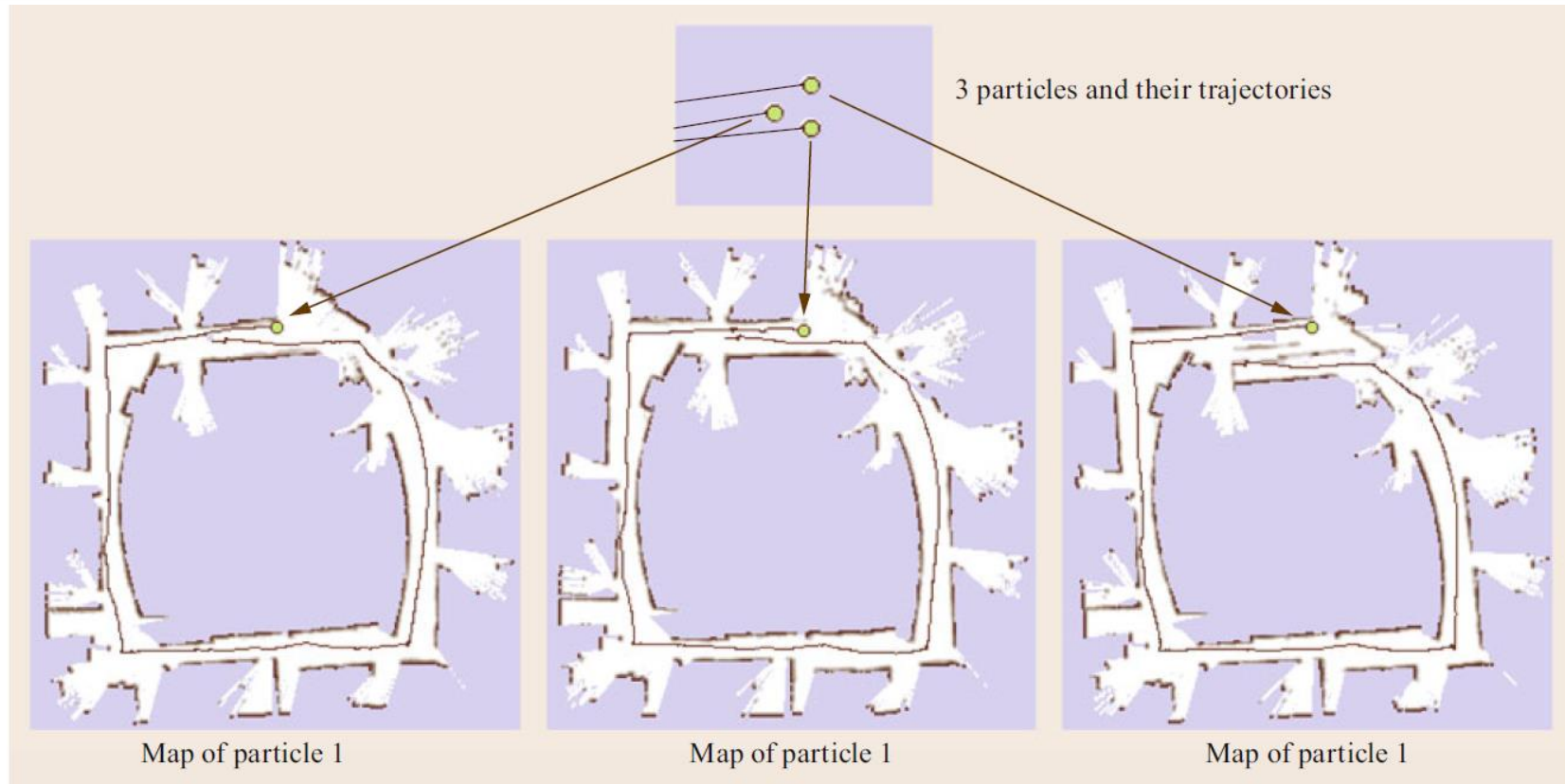
FastSLAM : A Factored Solution to the SLAM problem with unknown data association, Michael Montemerlo, Thèse de doctorat, 2003.

# Varia

---

- Nous avons fait l'hypothèse que le data association mesure-landmark était connu
- S'il est inconnu, risque de se tromper
  - EKF-SLAM : on jette l'information, on ne pourra pas retourner en arrière pour modifier ☹️
  - FastSLAM : certaines particules feront les bonnes associations, d'autres les mauvaises. Au final, les bonnes survivront

# Loop-closure naturel



Peut provoquer décimation des particules  
Pas totalement robuste

Tiré de : Handbook of Robotics, 2016.

# SLAM monoculaire (MonoSLAM)

- 1 caméra + FastSLAM + SIFT
- Permet de localiser en temps réel la caméra et créer carte de l'environnement en 3D

Combining Localization with Recognition  
for Scene Augmentation using a  
Wearable Camera



Robert Castle, Georg Klein, David W. Murray  
Active Vision Laboratory,  
University of Oxford, UK  
<http://www.robots.ox.ac.uk/ActiveVision>

<http://www.doc.ic.ac.uk/~ajd/>

*R. Castle, G. Klein, D. W. Murray, Active Vision Lab, Université Oxford*



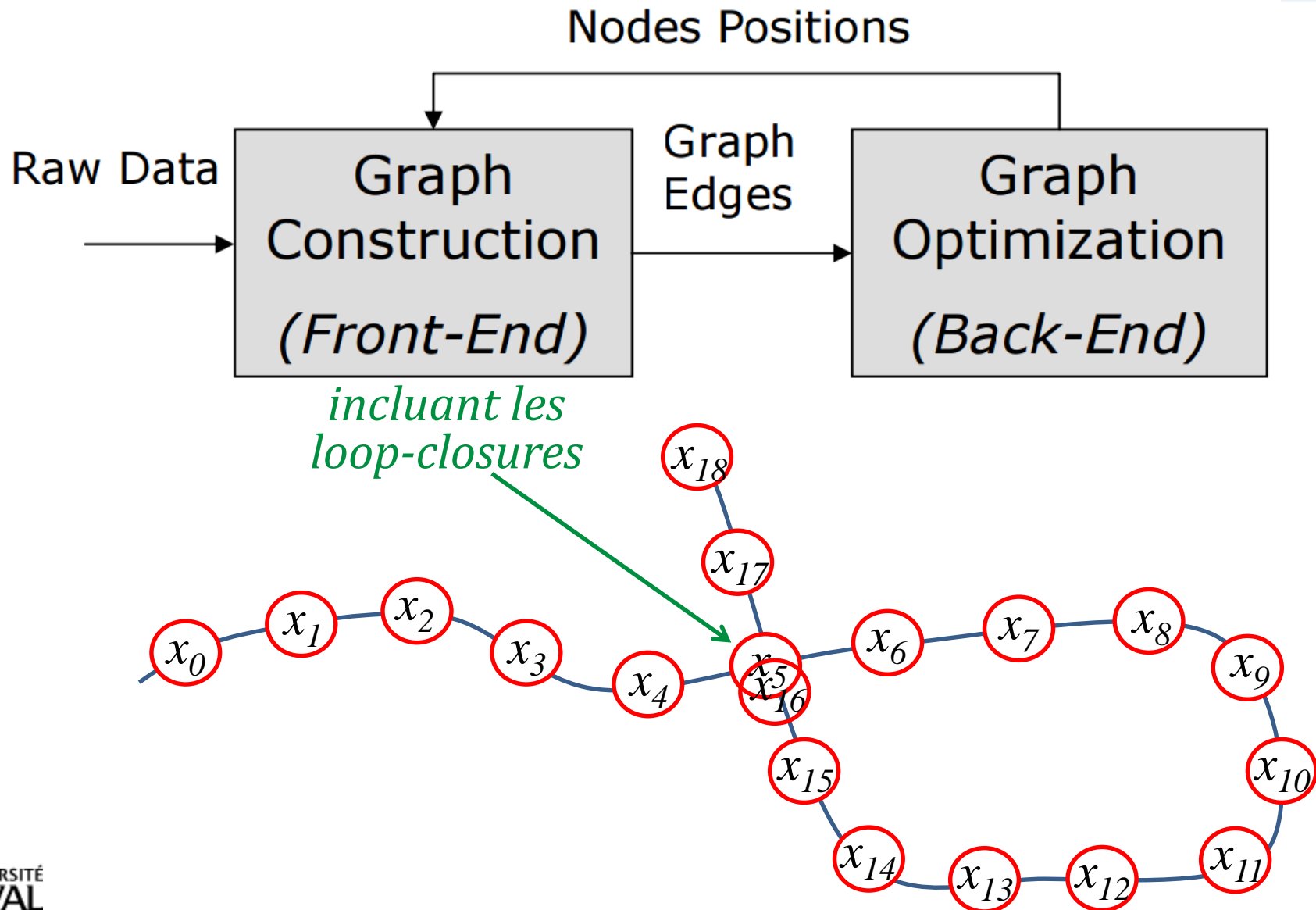
# GRAPH-SLAM

# Graph-SLAM

---

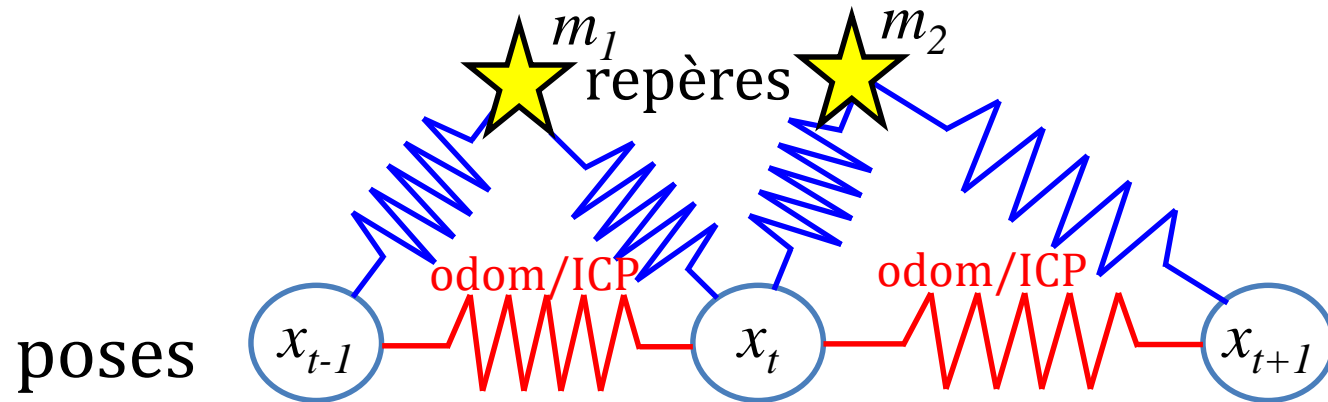
- Intuition : transformer le problème en graphe
- Nœuds : poses ou points de repères
- Arrêtes : contraintes non-linéaires (mesures ou odométries)

# Pose-Graph SLAM



# Graph SLAM Back-end : minimisation énergie

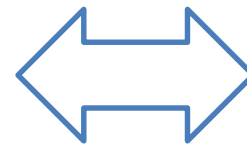
- Problème similaire à une structure élastique



*solution : max  
vraisemblance*

$$\operatorname{argmax}_{x_{1:t}, m} \left( p(x_{1:t}, m \mid u_{1:t}, z_{1:t}) \right) = \operatorname{argmax}_{x_{1:t}, m} \left( \prod_t p(x_t \mid x_{t-1}, u_t) p(z_t \mid x_t, m) \right)$$

$$= \operatorname{argmax}_{x_{1:t}, m} \left( \sum_t \log p(x_t \mid x_{t-1}, u_t) + \sum_t \log p(z_t \mid x_t, m) \right)$$



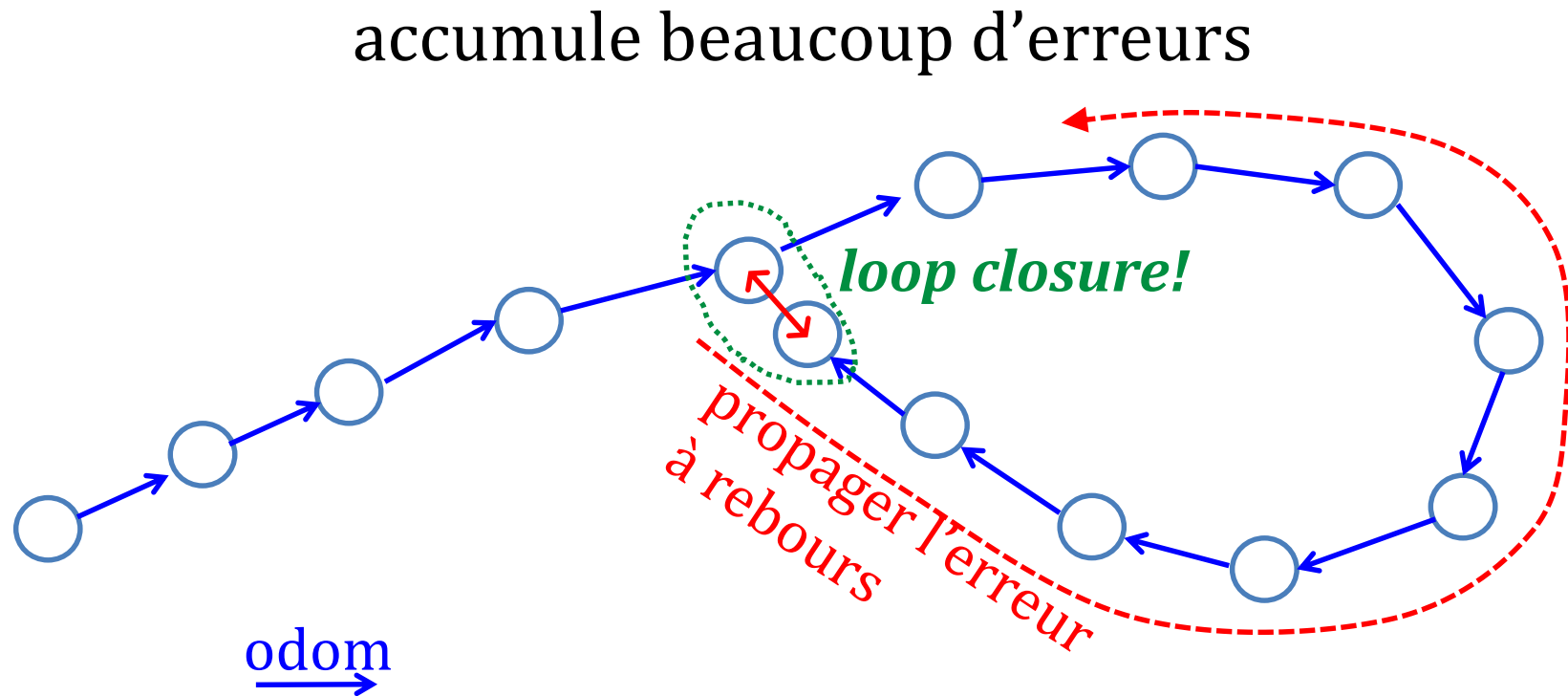
**énergie ressort**

$$\min \sum_i \frac{1}{2} k (x_i - x_{i0})^2$$

$$\log \left( \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \right) = \text{constante} - \frac{(x-\mu)^2}{2\sigma^2}$$

Rigidité d'un « ressort » =  $\frac{1}{\sigma^2}$  = inform.

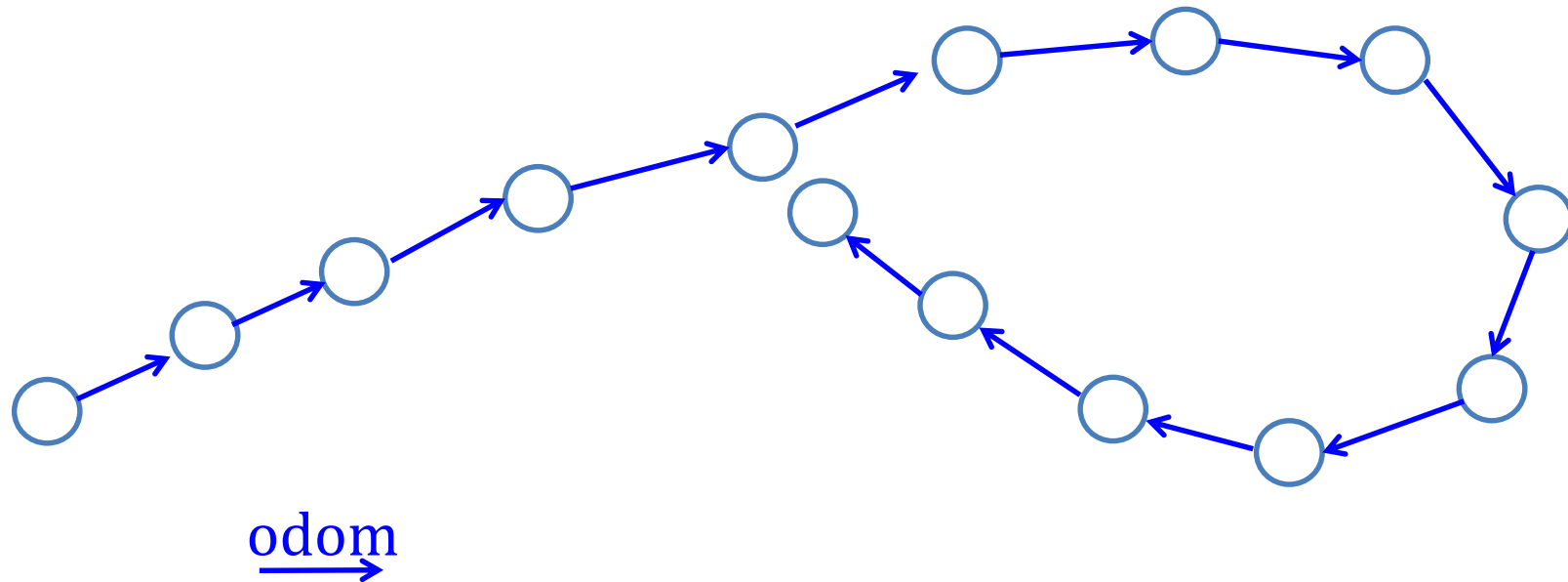
# Exemple Graph-SLAM online



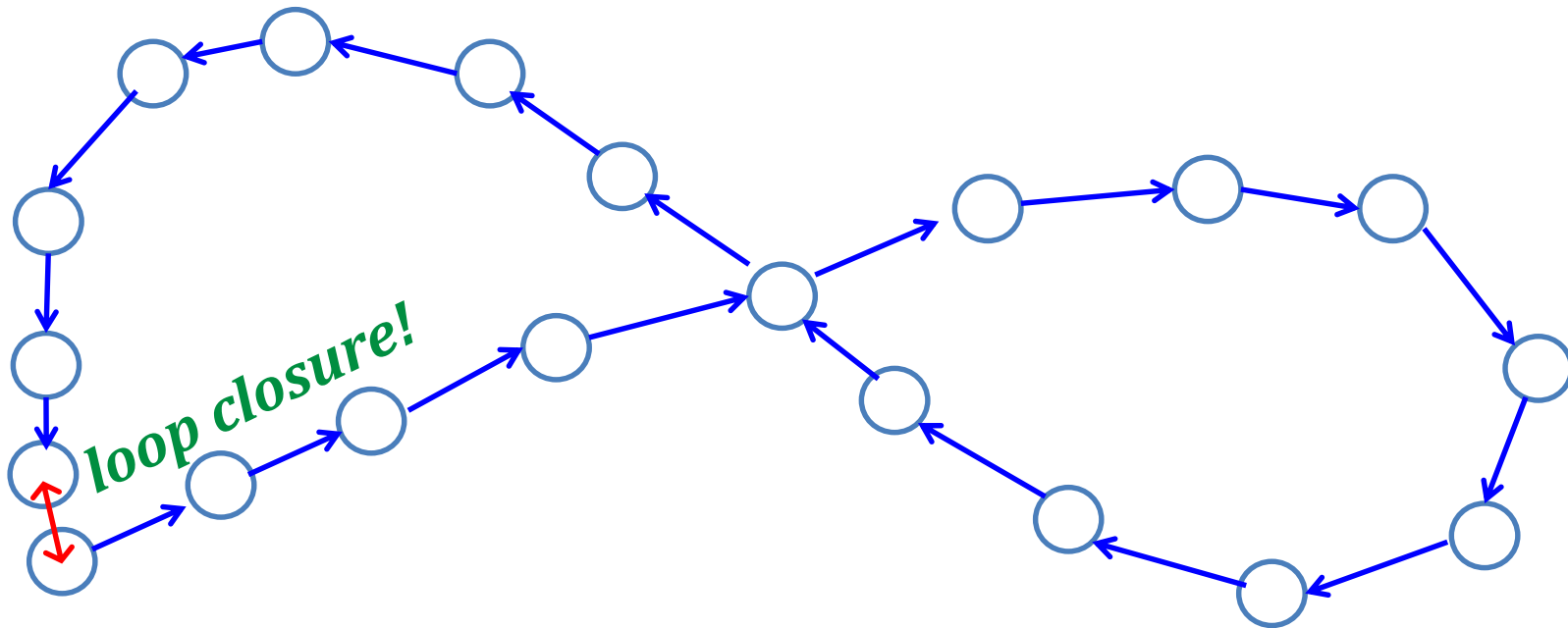
# Exemple Graph-SLAM online

## optimisation pose-graph

*(global relaxation, Back-End)*



# Exemple Graph-SLAM online



# Composantes Graph-SLAM

FRONT-END

- Odométrie (roues, visuelle)
  - possiblement raffinée par ICP
- Mesures de points de repères
- *Loop-closure*
  - détecter les fermetures de boucles sans erreur positive (catastrophiques)

BACK-END

- Optimisateur non-linéaire graphe
  - $g^2o$  (linéarise + Gauss-Newton, le plus populaire)  
 $g^2o$ : A General Framework for Graph Optimization, Kümmerle *et al.* ICRA 2011.
  - square SAM (et ses variantes) (*F. Daellert*)
  - TORO, etc.



# Trajectoire est importante!

---

- Beaucoup de *loop-closures*, et des mesures précises



- **Structure plus « rigide » == SLAM plus précis!**
- **Mauvais *loop-closures* == catastrophe!**
- **Grandes loop == fonc. objec. moins linéaire**  
(vitesse de convergence graph)

# Exemple

## Robust Loop Closing over Time for Pose Graph SLAM

Instituto de Investigación en  
Ingeniería de Aragón (I3A)

Yasir Latif: [ylatif@unizar.es](mailto:ylatif@unizar.es)  
José Neira: [jneira@unizar.es](mailto:jneira@unizar.es)

Volgenau School of Engineering  
George Mason University

César Cadena: [ccadenal@gmu.edu](mailto:ccadenal@gmu.edu)

*This work was supported by Spanish DPI2009-13710 and DPI2009-07130,  
by DGA-FSE (group T04), and by the US Army Research Office  
(W911NF-1110476)*