



UNIVERSITÉ
LAVAL

GLO-4001/7021

INTRODUCTION À LA ROBOTIQUE

MOBILE

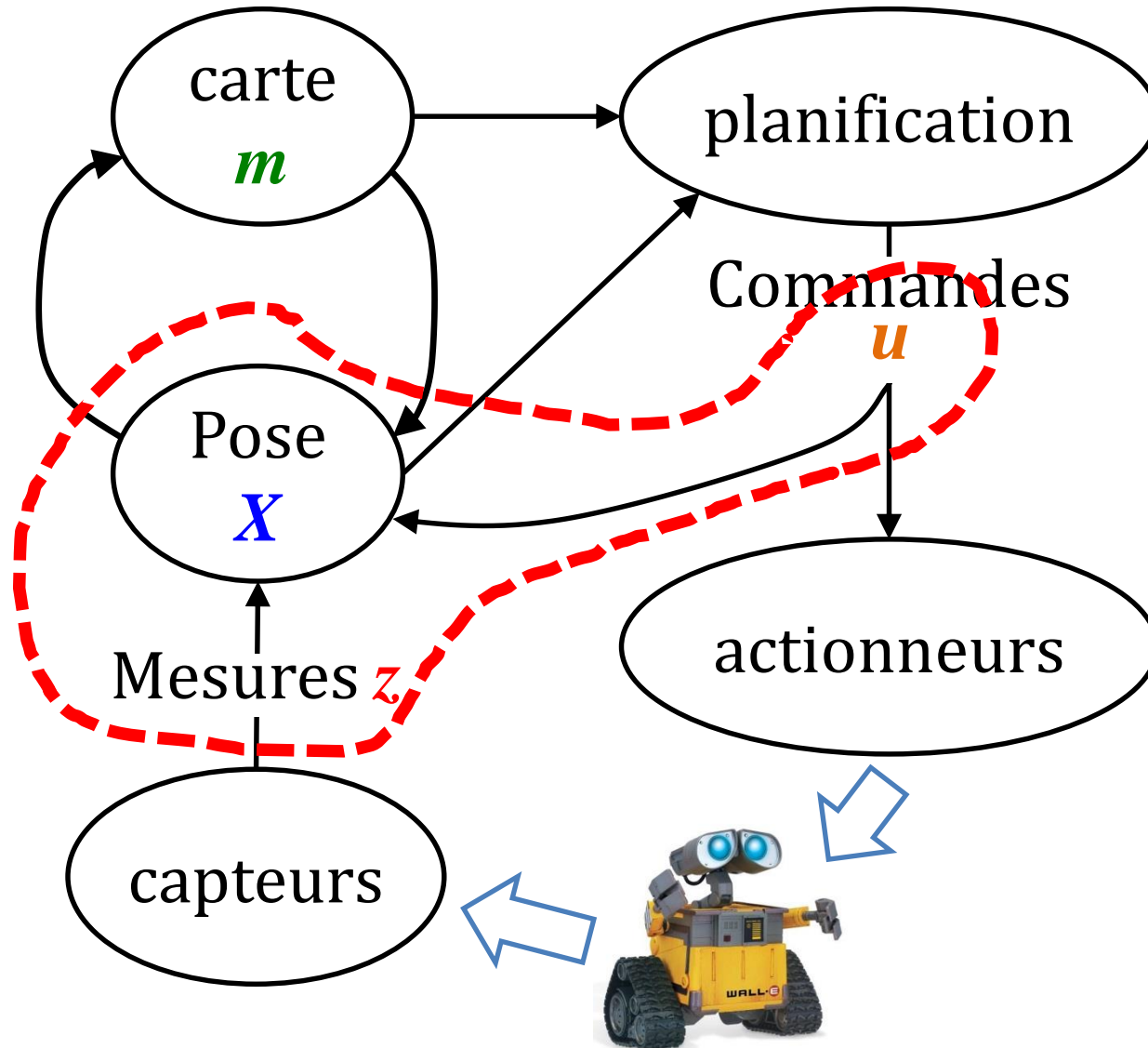
Filtres à particules

ICP

GLO-7021 : Présentations orales

- Durée 15 minutes
- Critères d'évaluations sur le site
- <http://www2.ift.ulaval.ca/~pgiguere/cours/IntroRobotique/notes/Grille%20%C3%A9valuation%20raux.pdf>
 - Éviter trop d'équations, texte
 - Privilégier figures, explication haut-niveau
 - Take-home message
- Ce vendredi, de 8h30 à 12h30
- Local VCH-00212

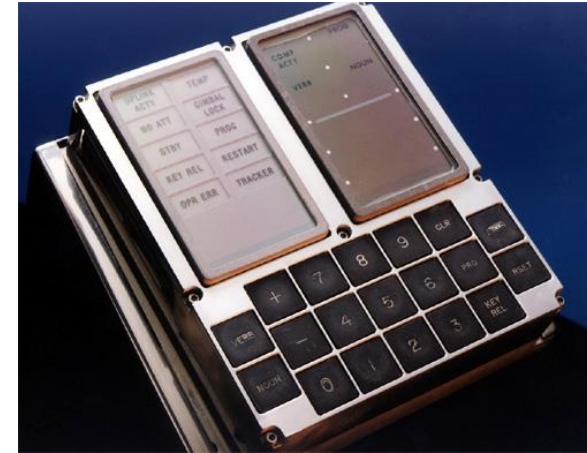
Feuille de route



Estimation d'état : filtres à particules

Estimation d'état : filtre Kalman

- Bruit gaussien
- Distribution unimodale
- Système linéaire :
 - filtre Kalman
- Système non linéaire :
 - filtre Kalman étendu (EKF)
 - filtre Kalman non-parfumé (UKF)
- EKF: rapide à effectuer
 - Apollo Guidance Computer (AGC)
 - 2.048 MHz
 - 2 048 x 16 RAM
 - 36 864 x 16 ROM



Interface DSKY du AGC



*ROM tressé à la main:
core rope memory*

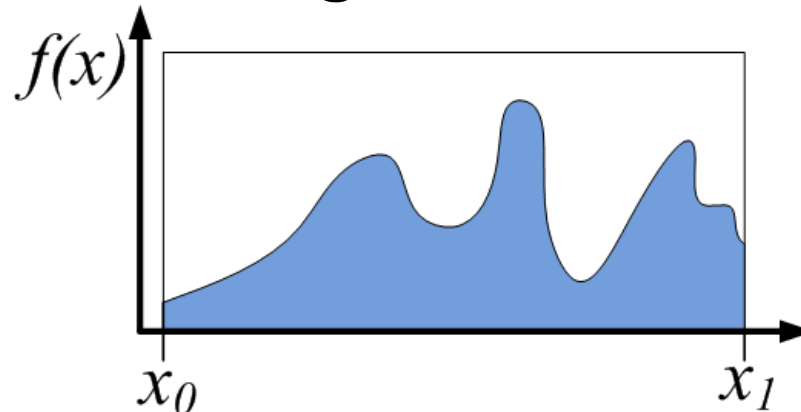
Méthodes Monte-Carlo

- Estimation de fonctions par méthodes aléatoires, par échantillonnage au hasard
- Développées durant le projet Manhattan, années 1940.



<http://hackaday.com/2015/09/11/fermiac-the-computer-that-advanced-the-manhattan-project/>

- Pour problèmes multi-variables ou sans solutions analytiques
- Exemple : calcul d'intégrale définie



Filtrage Bayésien (rappel)

```
Algorithme FiltreBayes ( $bel(x_{t-1}), u_t, z_t$ )
  for all  $x_t$  do
    Prédiction  $\longrightarrow$   $bel_{pred}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$  (1)
    Mise-à-jour  $\longrightarrow$   $bel(x_t) = \eta p(z_t | x_t) bel_{pred}(x_t)$  (2)
  endfor
  return  $bel(x_t)$ 
```

$bel()$: *belief* ou croyance (raccourci de notation)

u_t : commande au temps t

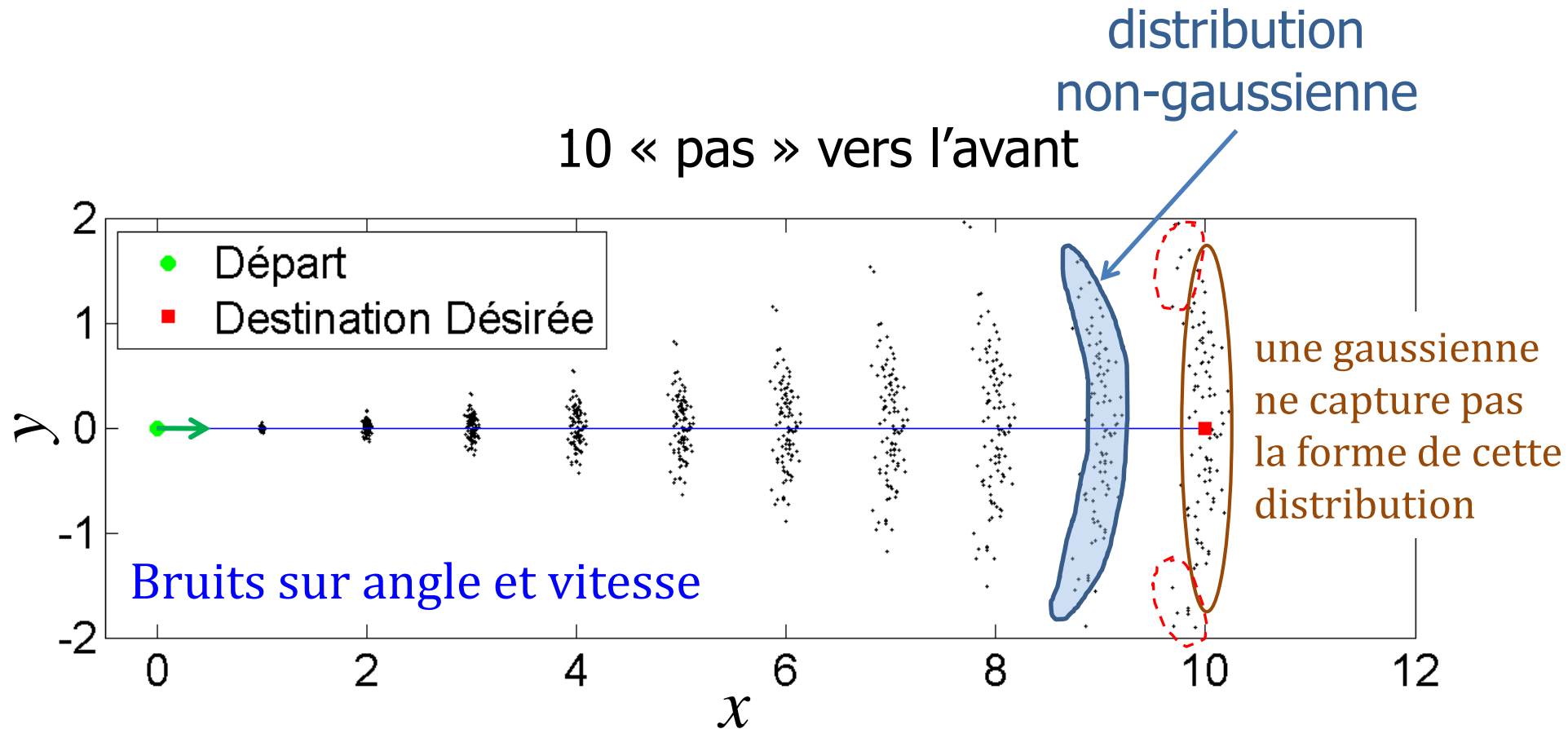
z_t : mesure au temps t , après la commande u_t

x_t : pose du robot au temps t , après u_t et z_t

η : facteur de normalisation

Difficile à implémenter exactement car pas toujours de solution analytique de l'intégrale (1)/produit (2)

Méthodes Monte-Carlo



100 simulations == 100 particules

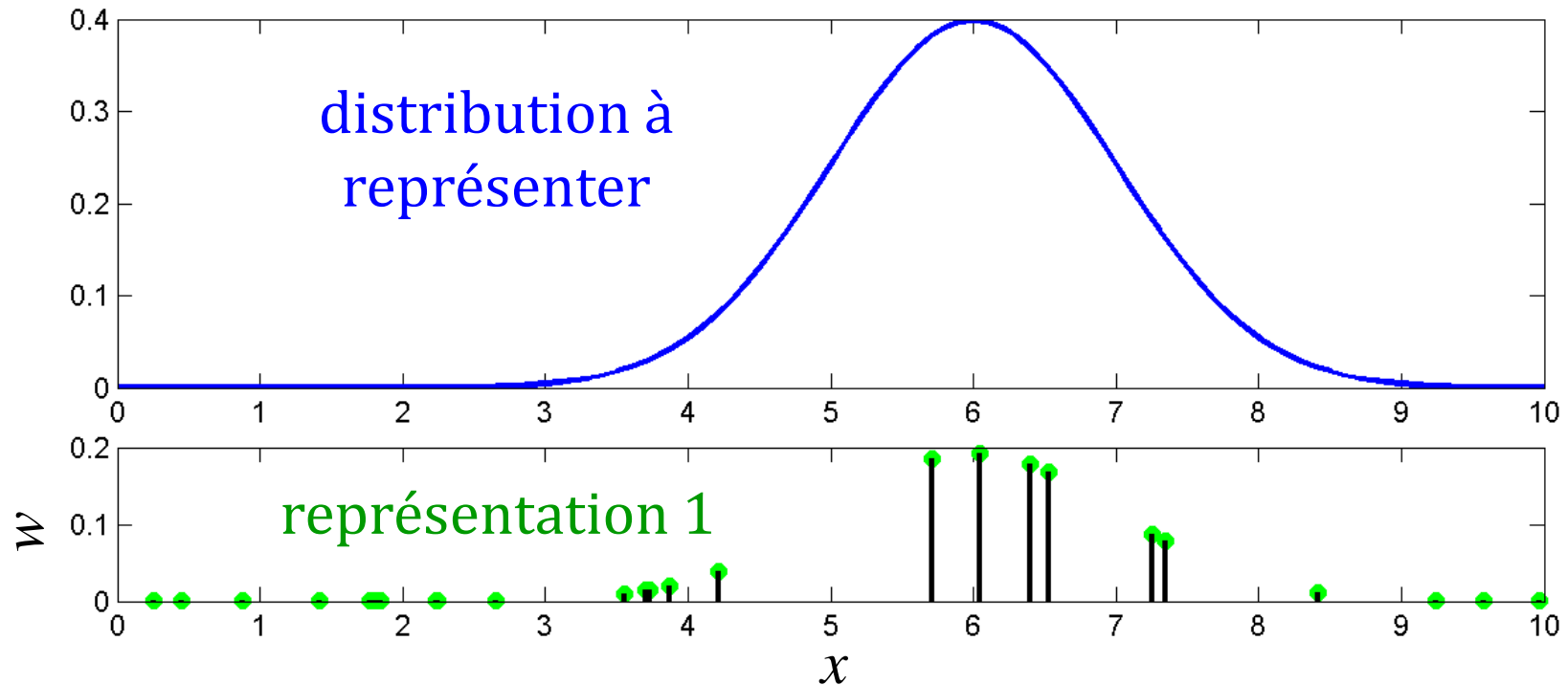
Monte-Carlo → filtre à particules

- Chaque particule représente une hypothèse

$i^{\text{ème}}$ particule : $X_i = [x \quad y \quad \theta]^T$ (pour un robot 2D)

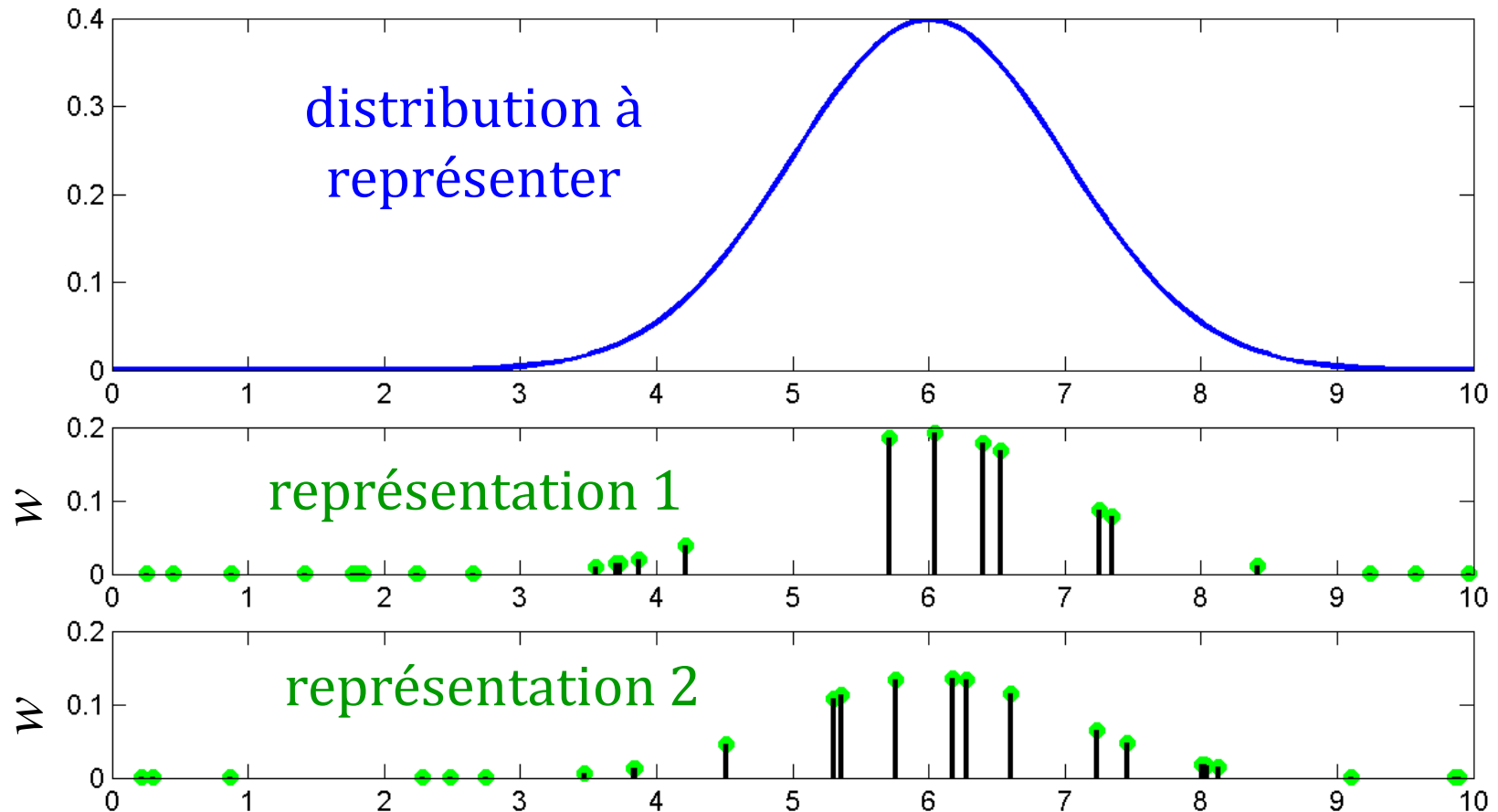
- Chaque particule a un poids associé w_i
- Le poids w_i représente en quelque sorte la « confiance » sur cette hypothèse
- Filtre utilise (en général) un nombre C fixe de particules

Exemple : distribution unimodale



- On pige une valeur x au hasard
- Le poids w est proportionnel à la hauteur de la distribution

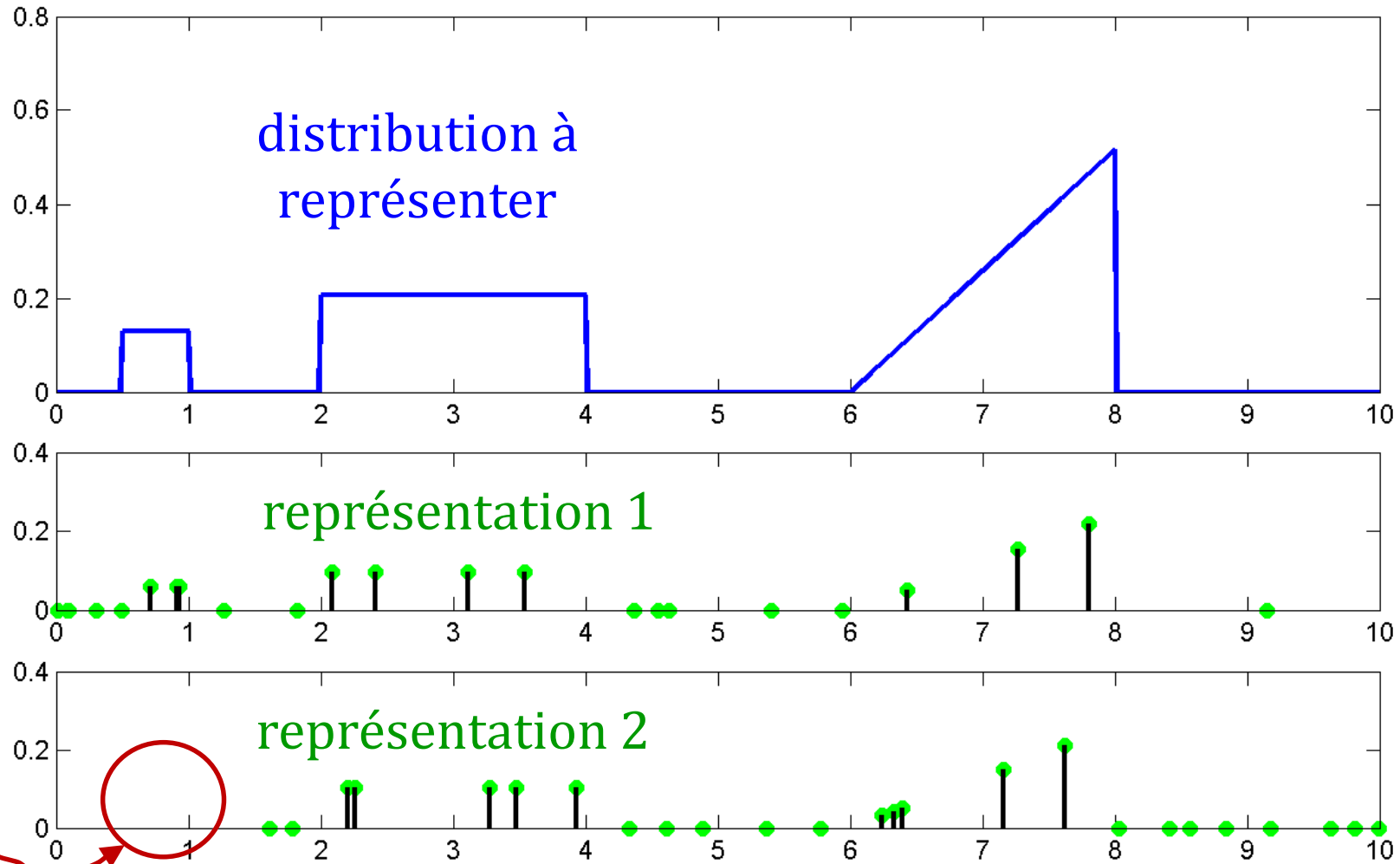
Exemple : distribution unimodale



Exemple : distribution multimodale

Non-gaussienne

passé à côté : rien n'est parfait...



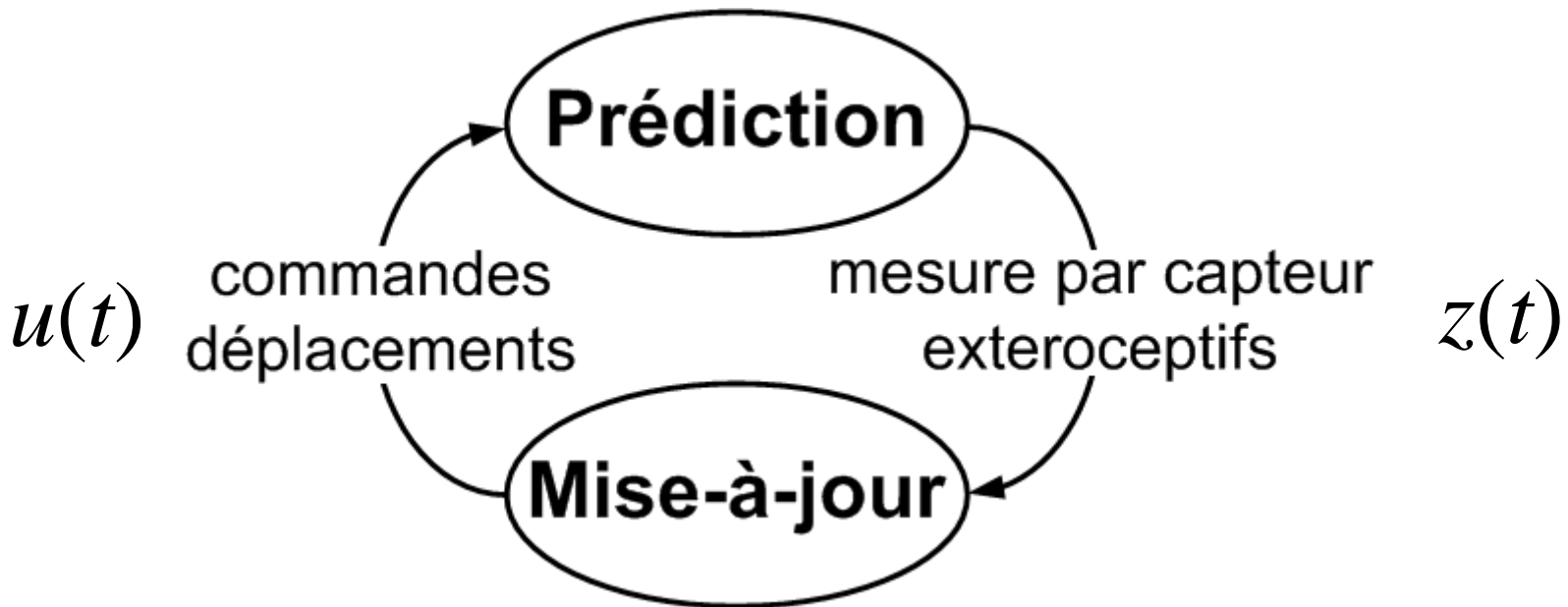
Variance \rightarrow étendu distribution

- On ne calcule plus la covariance P explicitement comme dans Kalman
 - Implicite par la distribution des particules
- Permet de représenter des distributions arbitrairement complexes



Filtre à particule

Intègre l'**information** en 2 étapes,
tout comme les filtres de *Kalman*



Filtre à particules : aperçu

```
while (explore)
```

```
  for i=1:C
```

```
     $X_i(k+1) = f_X(X_i(k), u(k), \sigma)$       Prédiction
```

on simule avec le bruit!

```
  end
```

```
   $z(k+1) = \text{mesure}()$  ;
```

```
  for i=1:C
```

```
     $w_i(k+1) = p(z(k+1) | X_i(k+1))w_i(k)$       Mise-à-jour
```

```
  end
```

```
  for i=1:C
```

```
     $w_i(k+1) = \frac{w_i(k+1)}{\sum_j \{w_j(k+1)\}}$       Normalisation
```

```
  end
```

```
   $N_{\text{eff}} = 1 / \sum_{i=1}^N w_i^2$ 
```

```
  if (Neff < Nseuil)
```

```
     $X_i(k+1) = \text{resample}(X_i(k+1), w_i(k+1))$  ;
```

Ré-échantillonnage

```
     $w_i(k+1) = 1/C$ 
```

```
  end
```

```
  k=k+1
```

```
end
```

Filtre à particules : aperçu

```
while (explore)
  for i=1:C
     $X_i(k+1) = f_X(X_i(k), u(k), \sigma)$  Prédiction
    on simule avec le bruit!
  end
   $z(k+1) = \text{mesure}()$ ;
  for i=1:C
     $w_i(k+1) = p(z(k+1) | X_i(k+1))w_i(k)$  Mise-à-jour
  end
  for i=1:C
     $w_i(k+1) = \frac{w_i(k+1)}{\sum_j \{w_j(k+1)\}}$  Normalisation
  end
   $N_{\text{eff}} = 1 / \sum_{i=1}^N w_i^2$ 
  if (Neff < Nseuil)
     $X_i(k+1) = \text{resample}(X_i(k+1), w_i(k+1))$ ; Ré-échantillonnage
     $w_i(k+1) = 1/C$ 
  end
  k=k+1
end
```


Étape 1 : prédiction du F. P.

- On déplace la particule selon :

- le modèle du système,
- les commandes $u(k)$,
- le **bruit v**

$$X_i(k+1) = f_X(X_i(k), u(k), \sigma_V)$$

pas besoin d'avoir $f_X(\bullet)$ linéaire...

- On répète pour chaque particule $i=1:C$
- Bref on déplace les C particules en simulant le robot **avec le bruit**

Pour notre robot 2D

$$\Delta d = (V + N(0, \sigma_V))\Delta t$$

$$f_x = x + \Delta d \cos \phi$$

$$f_y = y + \Delta d \sin \phi$$

$$f_\phi = \phi + \Delta t(\omega + N(0, v_\omega))$$

Étape 1 : prédiction, Kalman vs. F.P.

- Dans Kalman (étendu ou pas), on **ne propage pas le bruit sur la pose**

Kalman

$$\hat{x}(k+1|k) = \Phi \hat{x}(k) + \Gamma u(k)$$

Kalman étendu

$$\hat{X}(k+1|k) = f_X(\hat{X}(k), u(k))$$

- Dans filtre à particule, on propage avec du bruit

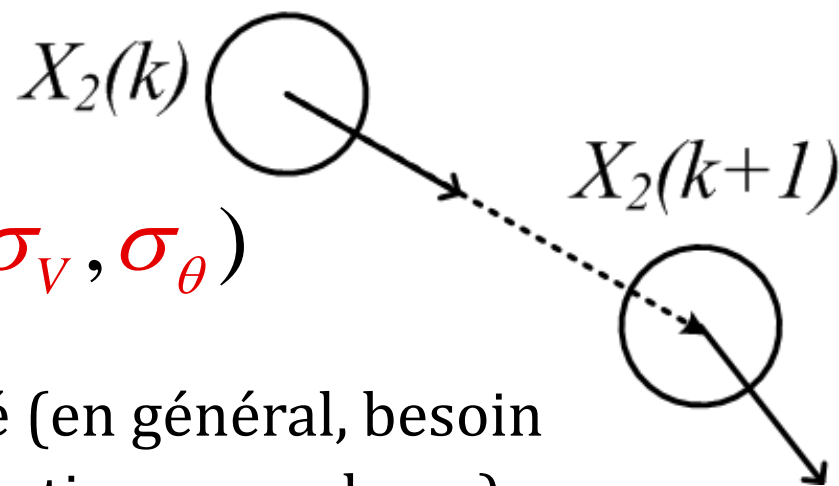
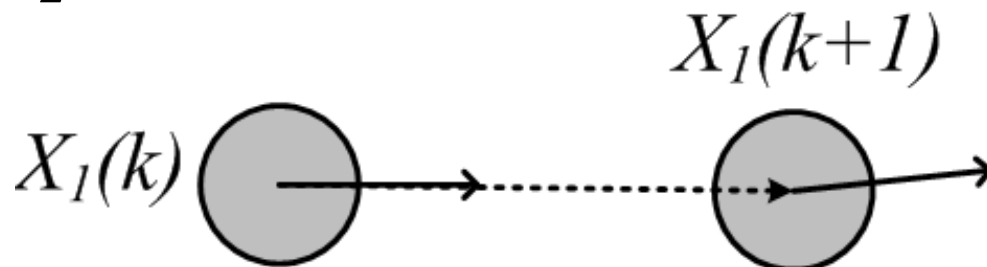
$$X_i(k+1) = f_X(X_i(k), u(k), \sigma_V)$$

- Et il n'y a pas de matrice de covariance P

~~$$P(k+1|k) = \Phi(k)P(k)\Phi(k)^T + C_v(k)$$~~

Filtre à particules : exemple prédiction

- Deux particules, X_1 et X_2
- Vitesse constante V
- Léger bruit sur angle
- Léger bruit sur vitesse



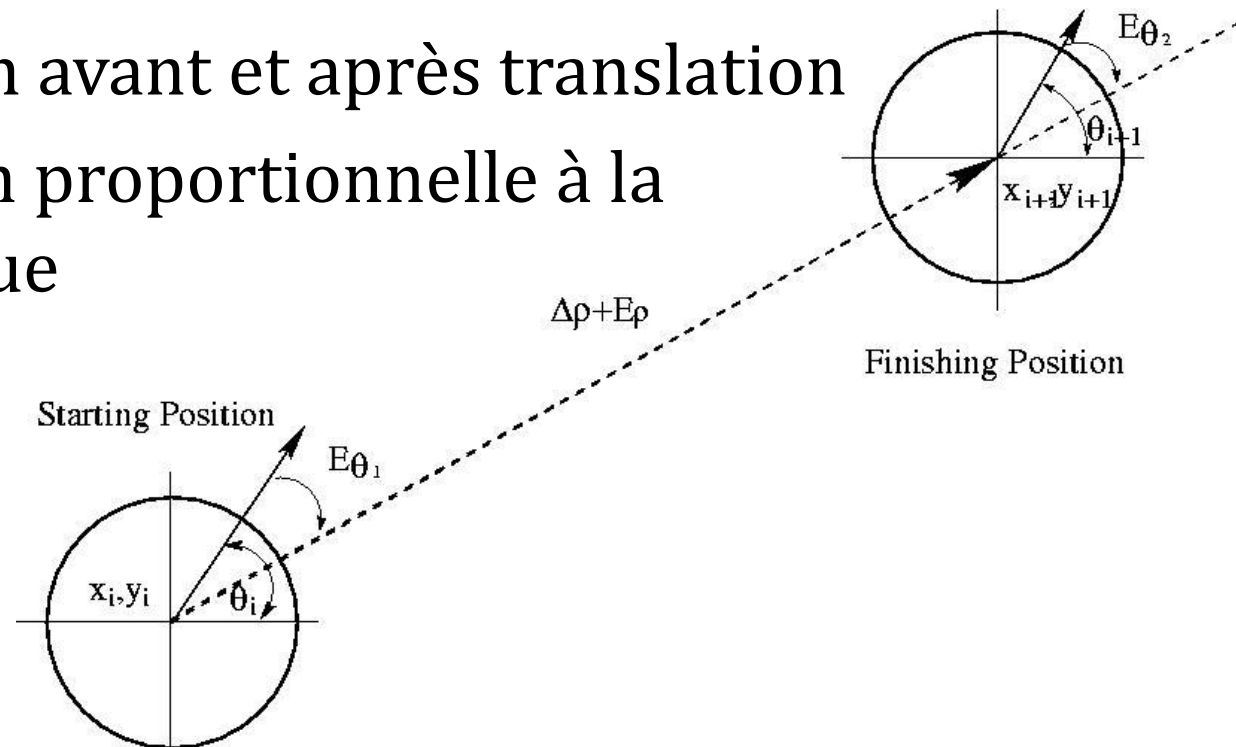
$$X_i(k+1) = f_X(X_i(k), u(k), \sigma_V, \sigma_\theta)$$

f_X peut être arbitrairement compliqué (en général, besoin de plus de particules pour les distributions complexes)

Autre exemple bruit pour un pas

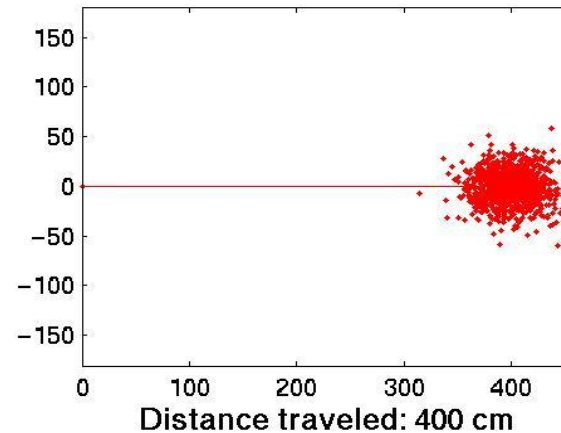
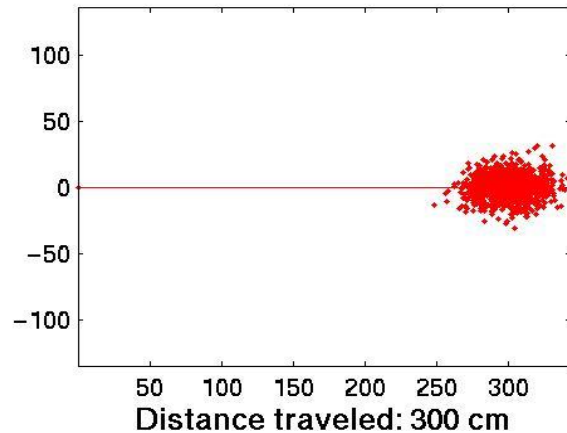
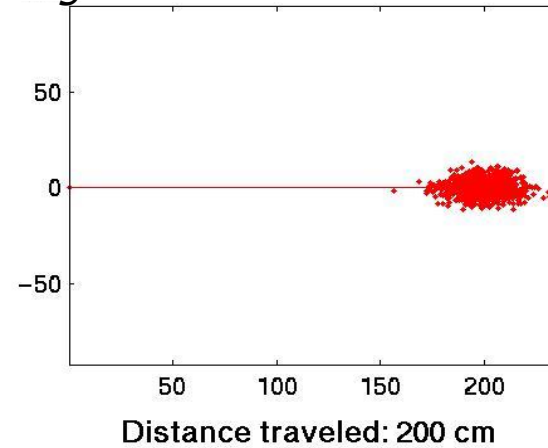
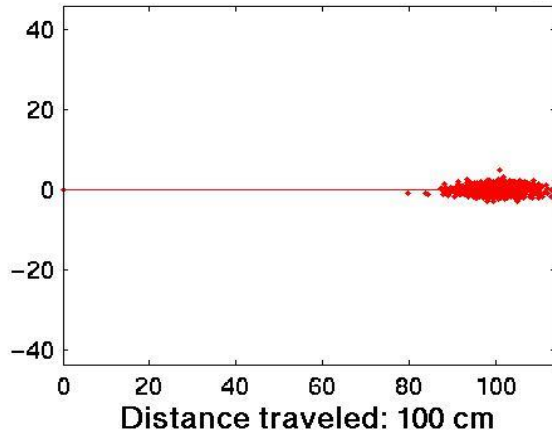
- Rotation : bruit gaussien
- Translation : bruit gaussien
- Un pas:
 - erreur de rotation avant et après translation
 - erreur translation proportionnelle à la distance parcourue

(permet de mieux décorréler x , y et θ car autant de sources de bruits que de degrés de liberté)



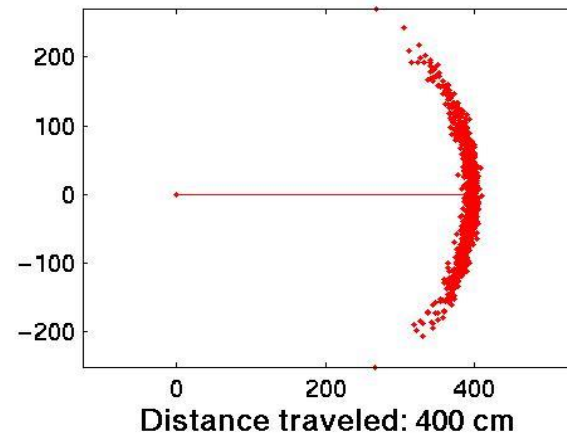
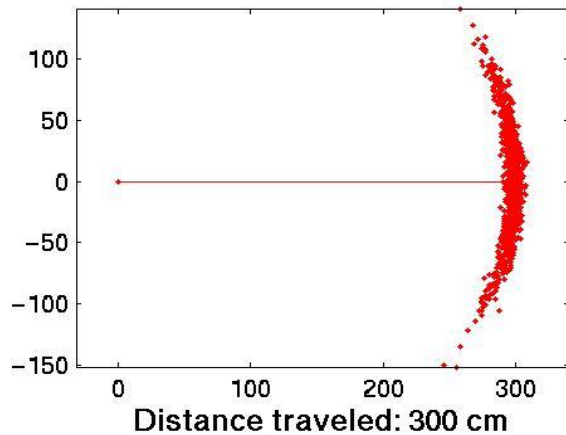
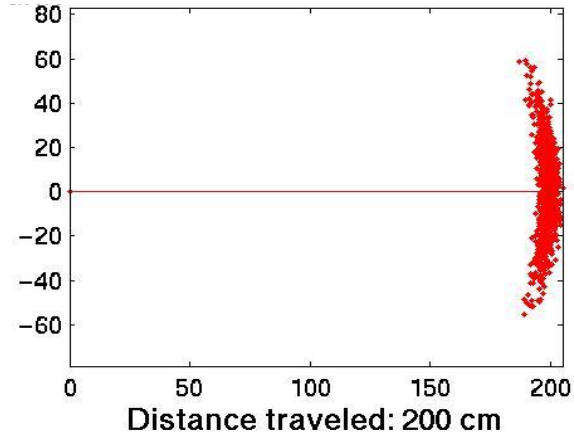
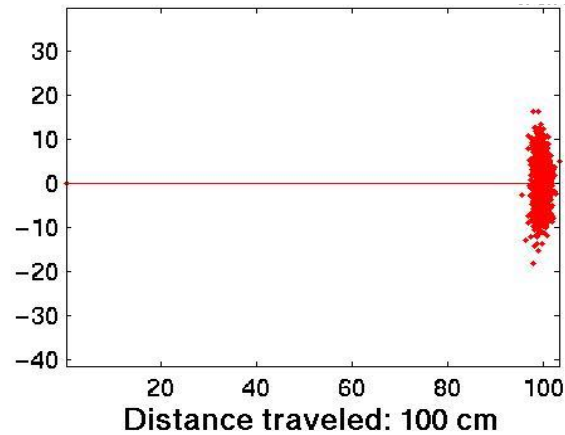
Modèle bruit déplacement

$$\sigma_{trans} = 5 \quad \sigma_{angle} = 1$$



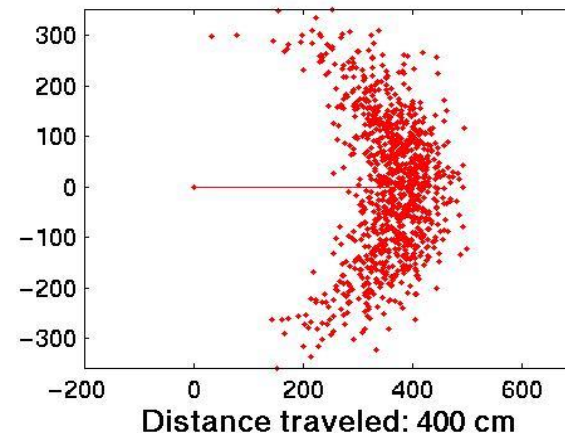
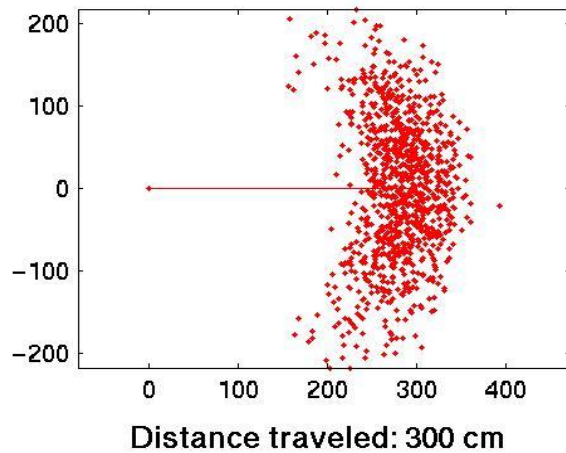
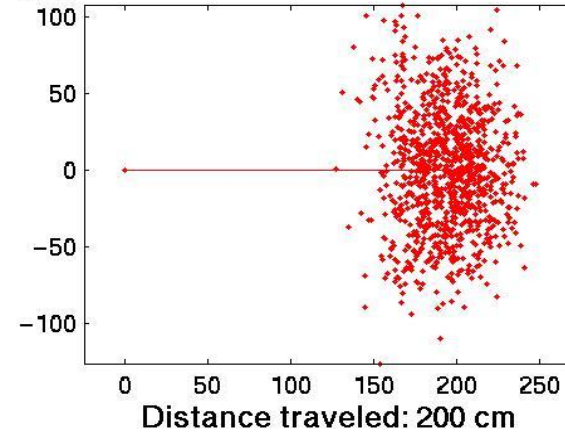
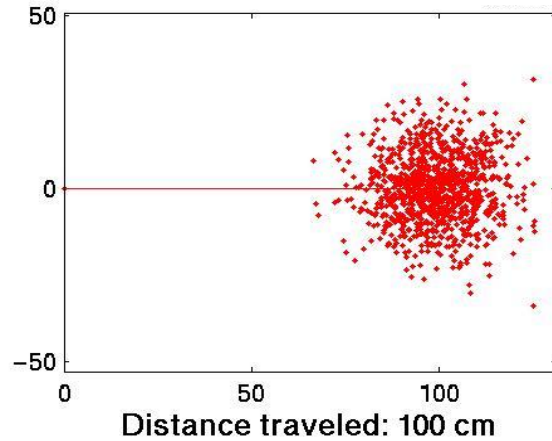
Modèle bruit déplacement

$$\sigma_{trans} = 1 \quad \sigma_{angle} = 5$$



Modèle bruit déplacement

$$\sigma_{trans} = 10 \quad \sigma_{angle} = 10$$



Bien modéliser le bruit de déplacement

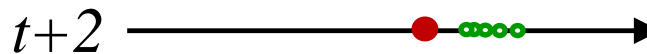
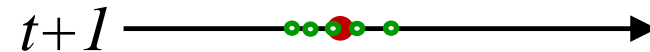
- Au pire, on surestime le bruit
 - sous-estimer est néfaste... (divergence du filtre à particule)



sous-estime le bruit

surestime le bruit

Au départ :



aucune particule près
de la position réelle



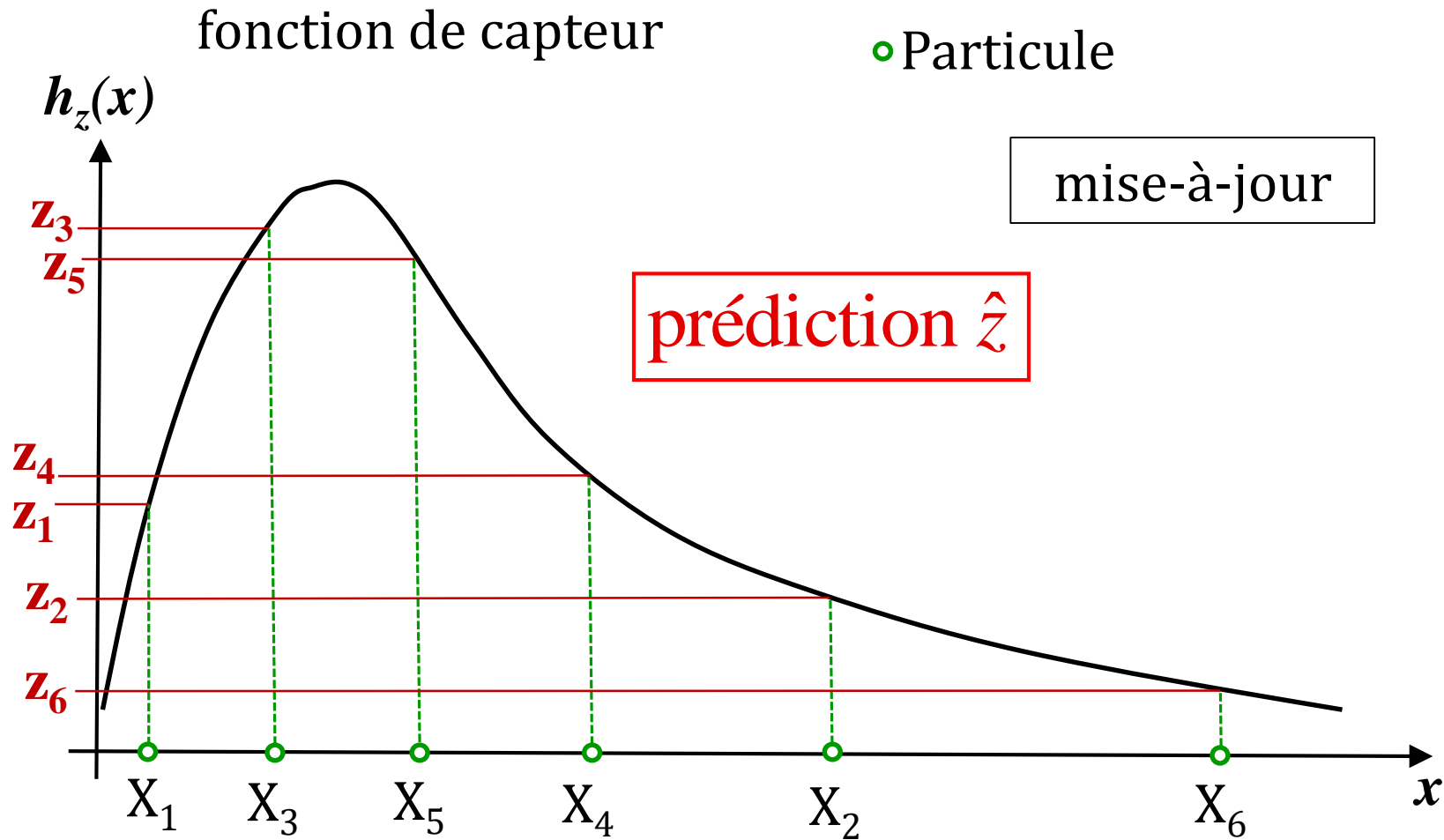
temps

Étape 2 : Mise-à-jour

- Incorporer l'information de capteurs extéroceptifs
- Réduire l'incertitude
- Utiliser la distribution de probabilité des mesures $h_z(X)$ pour ajuster le poids w_i de chaque particule

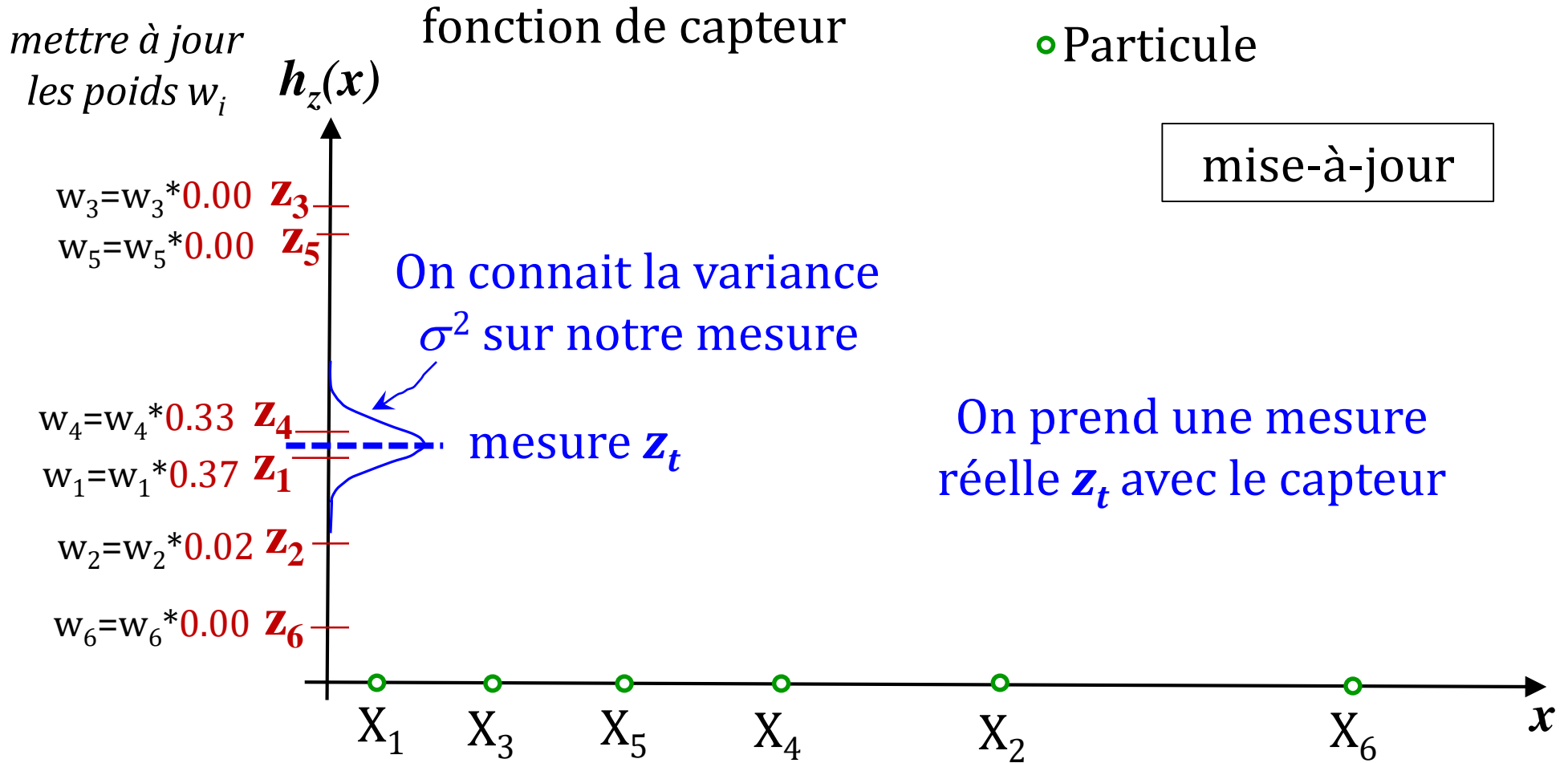
Exemple de filtre, fonction non-bijectionne

Prédire la mesure attendue pour
chaque particule X_i



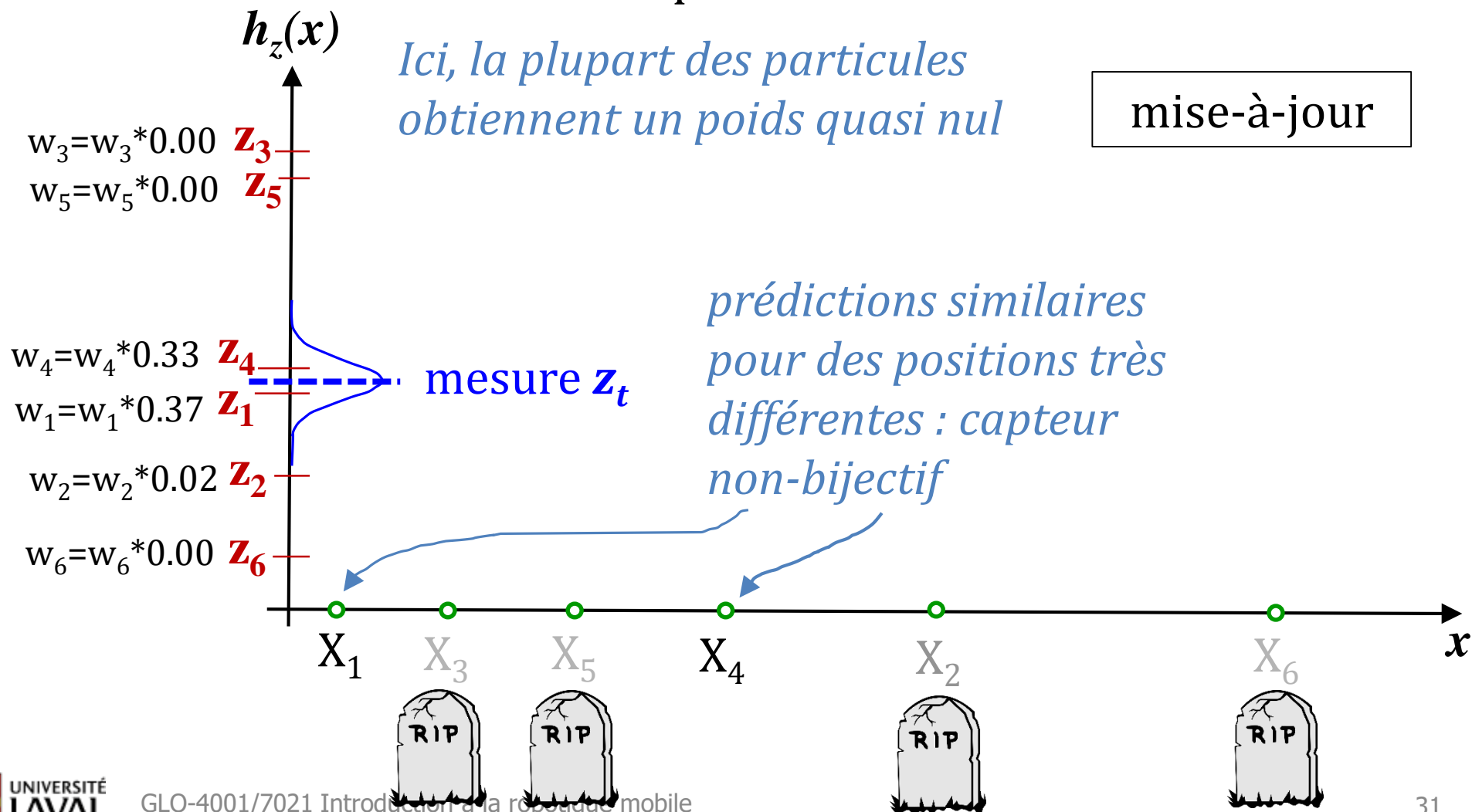
*particules X_i un peu
partout : je suis perdu!*

Exemple de filtre, fonction non-bijective

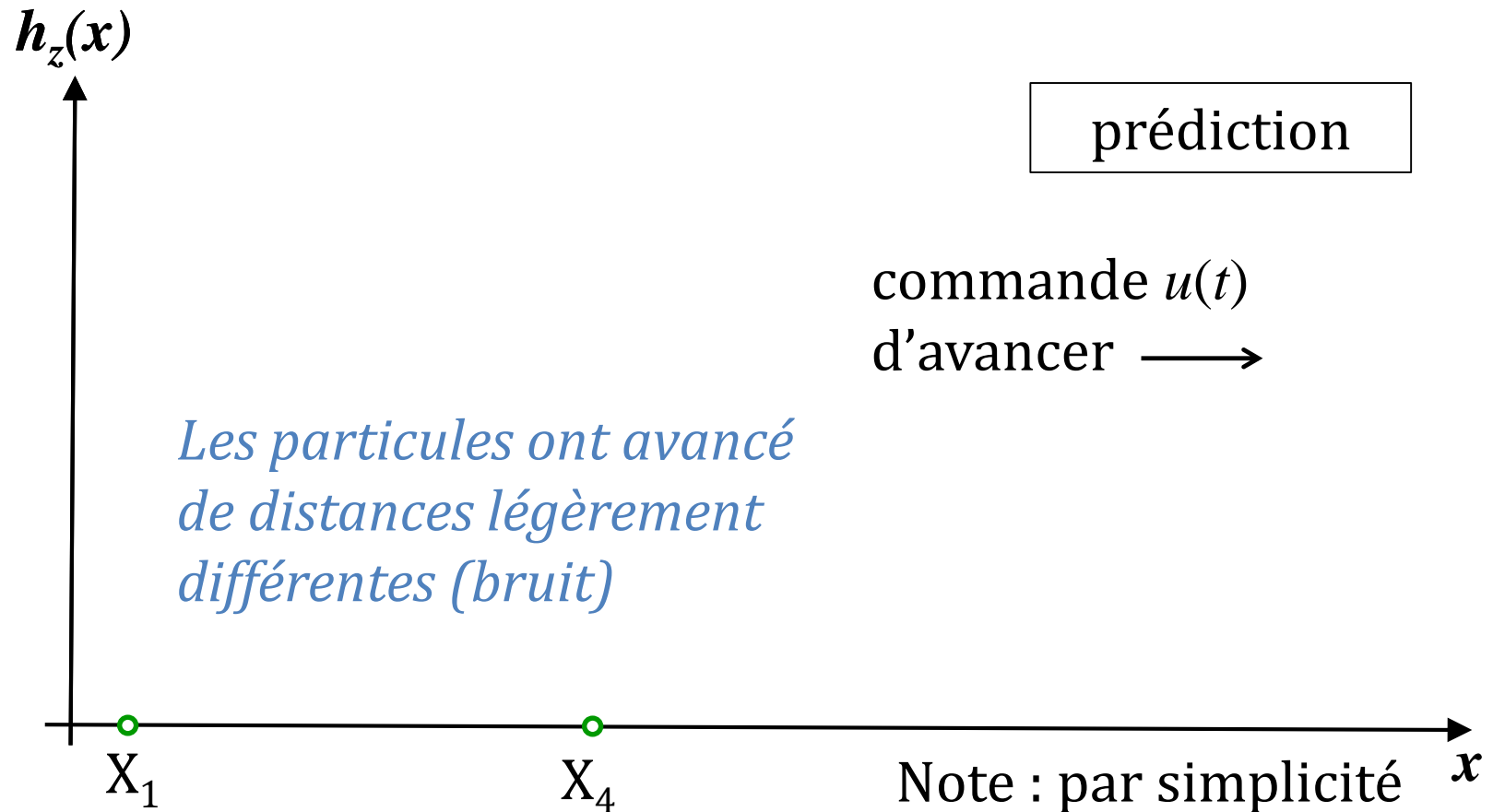


Exemple de filtre, fonction non-bijective

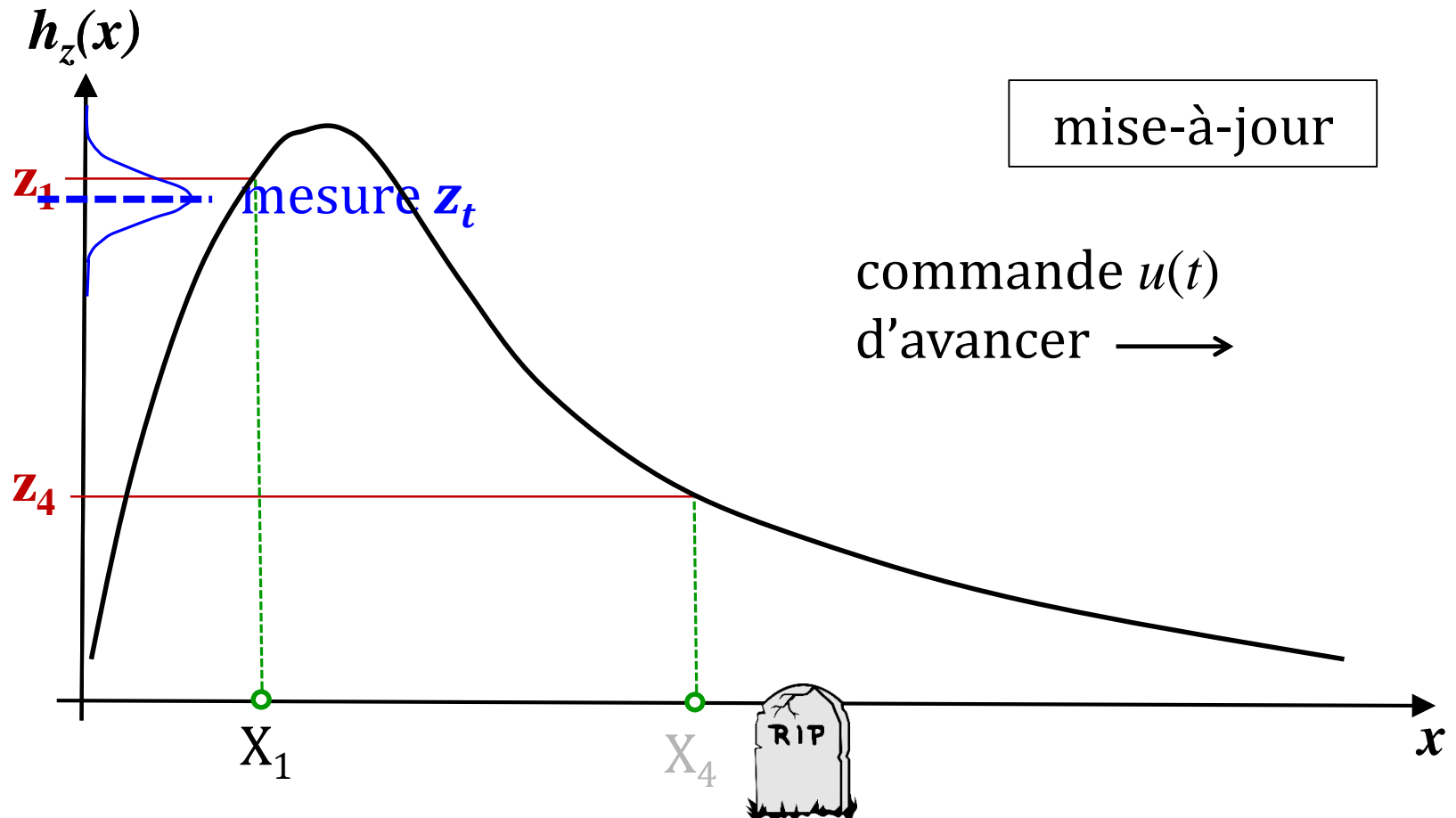
fonction de capteur



Exemple de filtre, fonction non-bijective



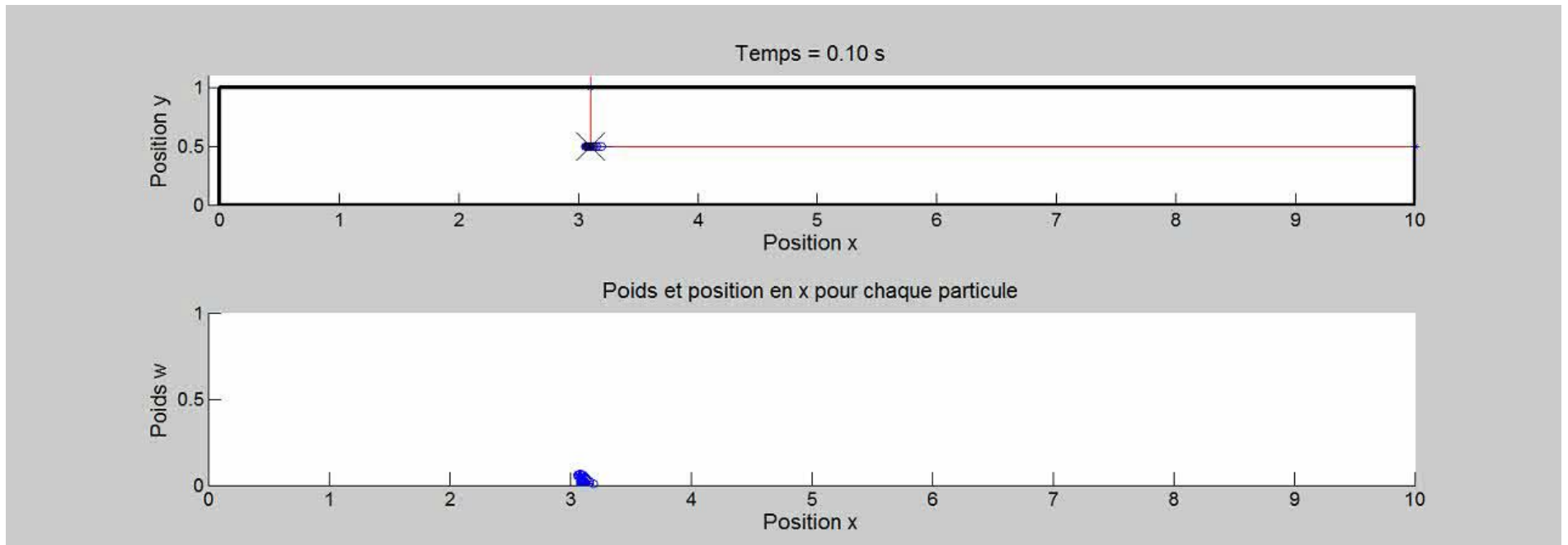
Exemple de filtre, fonction non-bijective



Le filtre à particule a donc réussi à isoler la bonne hypothèse dans cet exemple, après un seul pas!

Appauvrissement avec le temps

- Avec les mises-à-jour, certaines particules finissent par avoir un poids $w_i \approx 0$
- Perte du pouvoir de représentation — *mesure laser*



Sample Importance Resampling : SIR

- Critère d'appauvrissement :

(N_{eff} = nombre effectif de particules)

$$N_{eff} = \frac{1}{\sum_{i=1}^c w_i^2}$$

$$w_1 = \frac{1}{4}, w_2 = \frac{1}{4}, w_3 = \frac{1}{4}, w_4 = \frac{1}{4} \rightarrow \frac{1}{\sum_1^4 \left(\frac{1}{4}\right)^2}$$

Sample Importance Resampling : SIR

- Critère d'appauvrissement : $N_{eff} = \frac{1}{\sum_{i=1}^c w_i^2}$
(N_{eff} = nombre effectif de particules)

$$w_1 = \frac{1}{4}, w_2 = \frac{1}{4}, w_3 = \frac{1}{4}, w_4 = \frac{1}{4} \Rightarrow \frac{1}{\sum_1^4 \left(\frac{1}{4}\right)^2} = \frac{1}{\sum_1^4 \frac{1}{16}} = \frac{1}{\frac{4}{16}} = \frac{16}{4} = 4$$

Sample Importance Resampling : SIR

- Critère d'appauvrissement : $N_{eff} = \frac{1}{\sum_{i=1}^c w_i^2}$
(N_{eff} = nombre effectif de particules)

$$w_1 = \frac{1}{4}, w_2 = \frac{1}{4}, w_3 = \frac{1}{4}, w_4 = \frac{1}{4} \Rightarrow \frac{1}{\sum_1^4 \left(\frac{1}{4}\right)^2} = \frac{1}{\sum_1^4 \frac{1}{16}} = \frac{1}{\frac{4}{16}} = \frac{16}{4} = 4$$

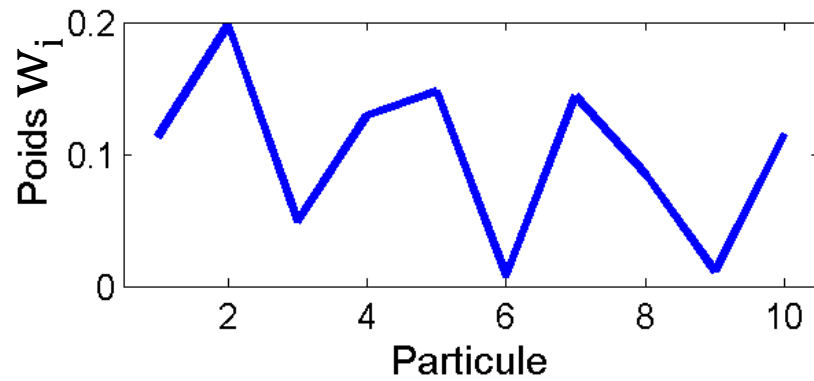
$$w_1 = 0.85, w_2 = 0.05, w_3 = 0, w_4 = 0.1 \Rightarrow N_{eff} = 1.3605$$

Sample Importance Resampling : SIR

- Si $N_{eff} < N_{seuil}$, il y a appauvrissement
 - peu de particules avec un poids w_i non-négligeable
- Normal, car à la longue l'état X_i diverge
- Doit donc rétablir une nouvelle population de particules
- Comment? En échantillonnant avec remplacement dans la population actuelle, avec probabilité w_i

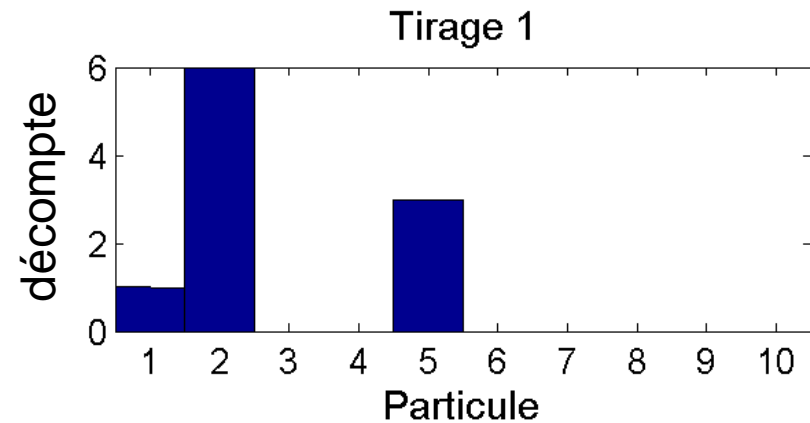
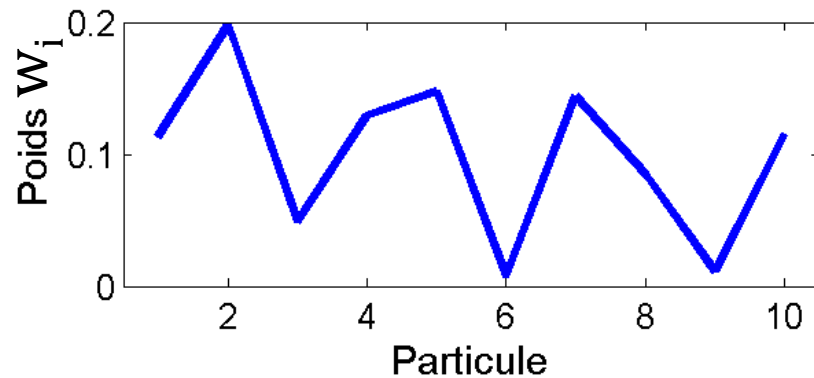
Sample Importance Resampling

- Exemple pour $C=10$ particules



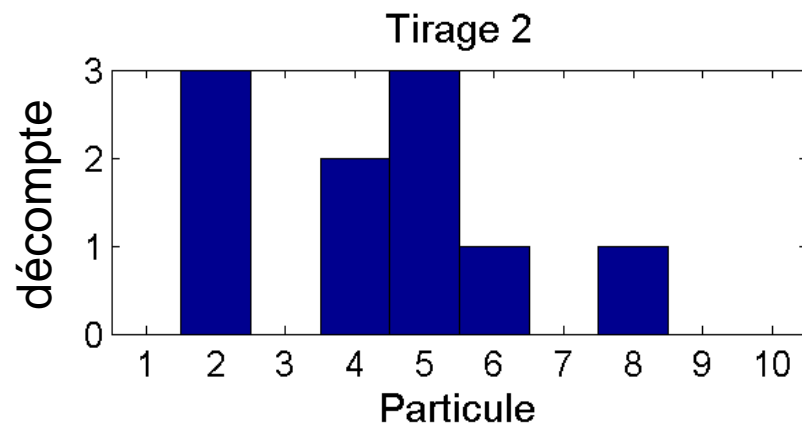
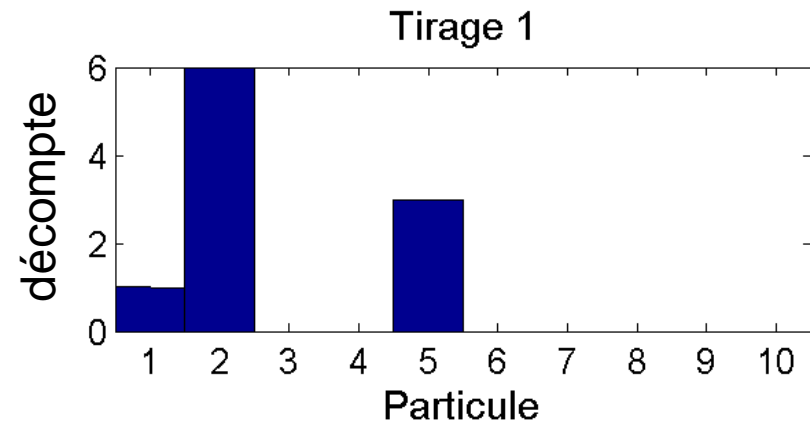
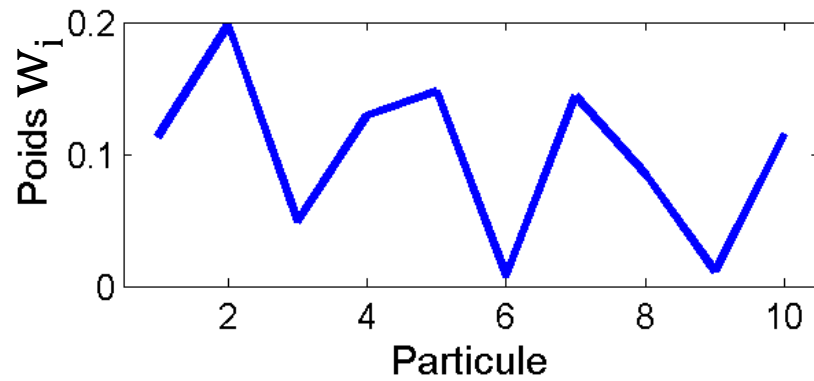
Sample Importance Resampling

- Exemple pour $C=10$ particules



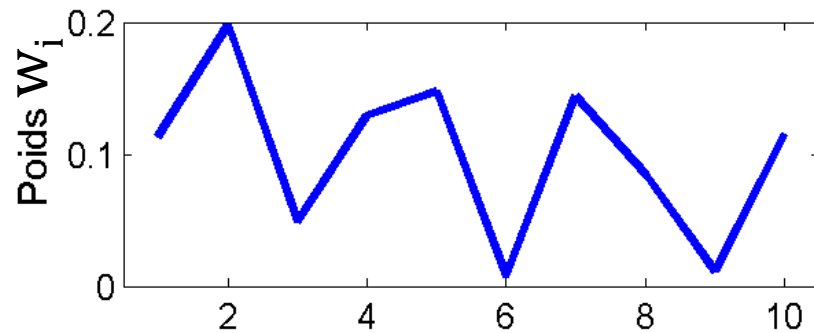
Sample Importance Resampling

- Exemple pour $C=10$ particules

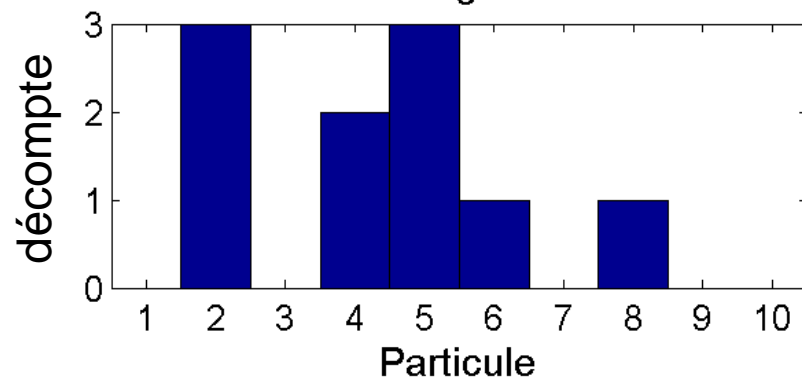


Sample Importance Resampling

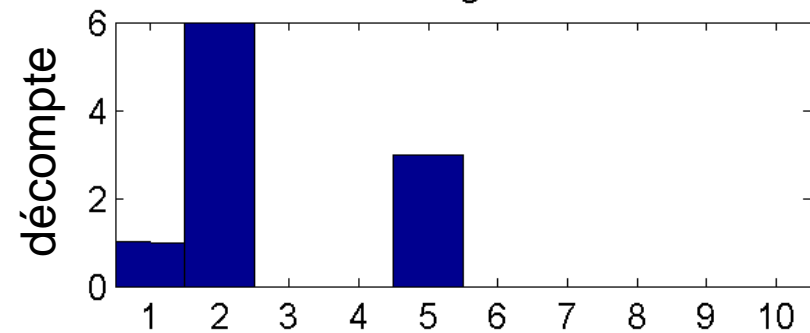
- Exemple pour $C=10$ particules



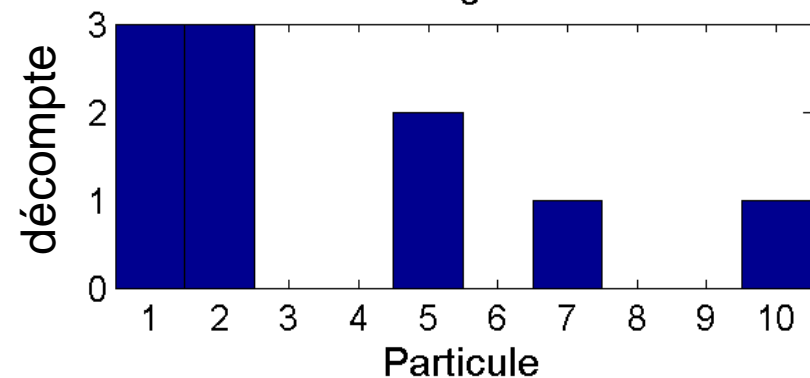
Tirage 2



Tirage 1

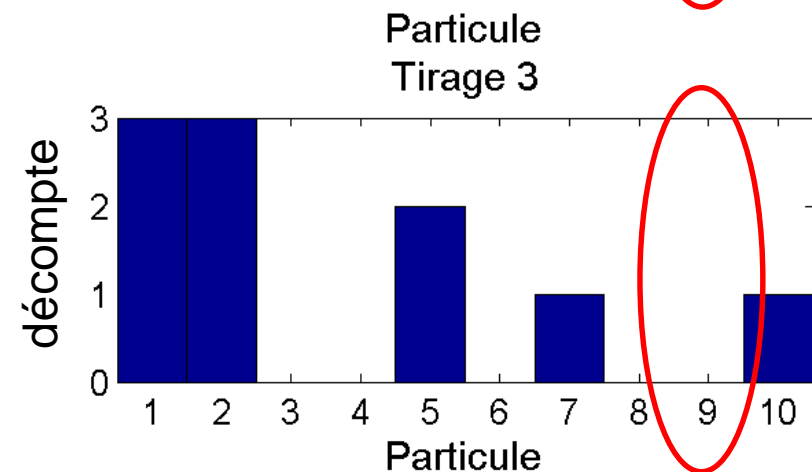
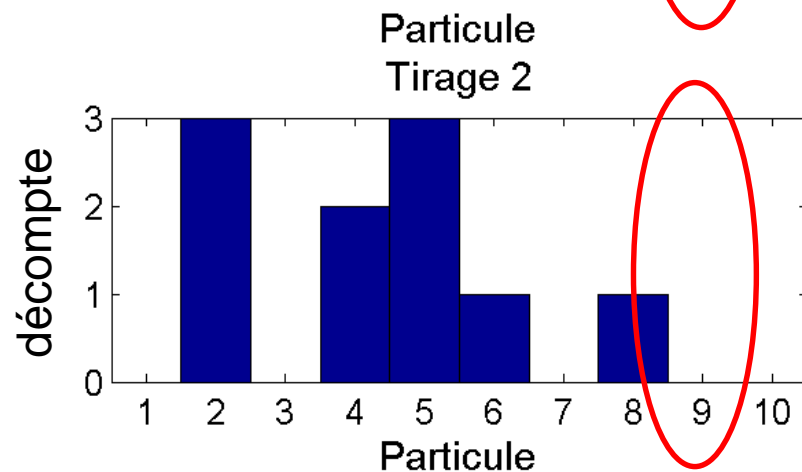
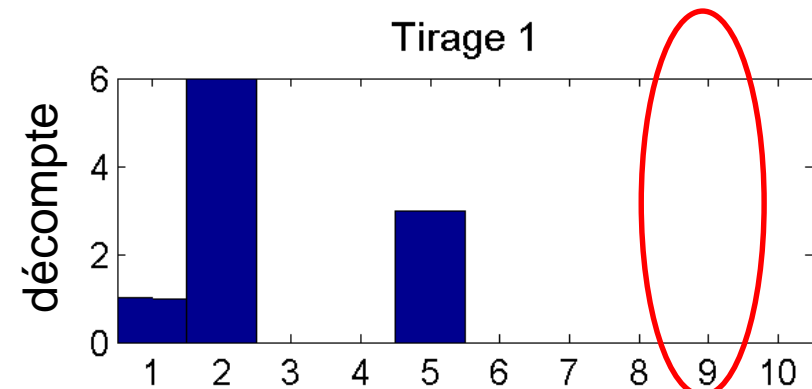
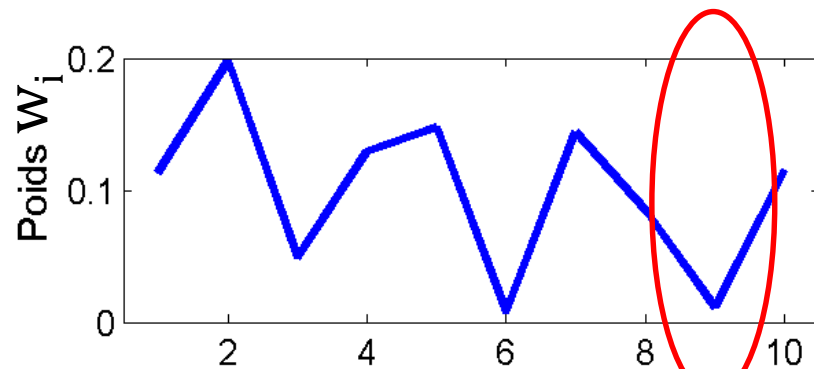


Tirage 3



Sample Importance Resampling

- Exemple pour $C=10$ particules



Particule 9 jamais pigée, car w_9 est très faible

Note : on ne fait qu'un seul tirage dans le filtre

Sample Importance Resampling : SIR

- Algorithme efficace de ré-échantillonnage :

Input: double $W[N]$

Require: $\sum_{i=1}^N W_i = 1$

$Q = \text{cumsum}(W)$; { calculate the running totals $Q_j = \sum_{l=0}^j W_l$ }

$t = \text{rand}(N+1)$; {t is an array of $N+1$ random numbers.}

$T = \text{sort}(t)$; {Sort them ($O(n \log n)$ time)}

$T(N+1) = 1$; $i=1$; $j=1$; {Arrays start at 1}

while ($i \leq N$) **do**

if $T[i] < Q[j]$ **then**

 Index[i]=j;

$i=i+1$;

else

$j=j+1$;

end if

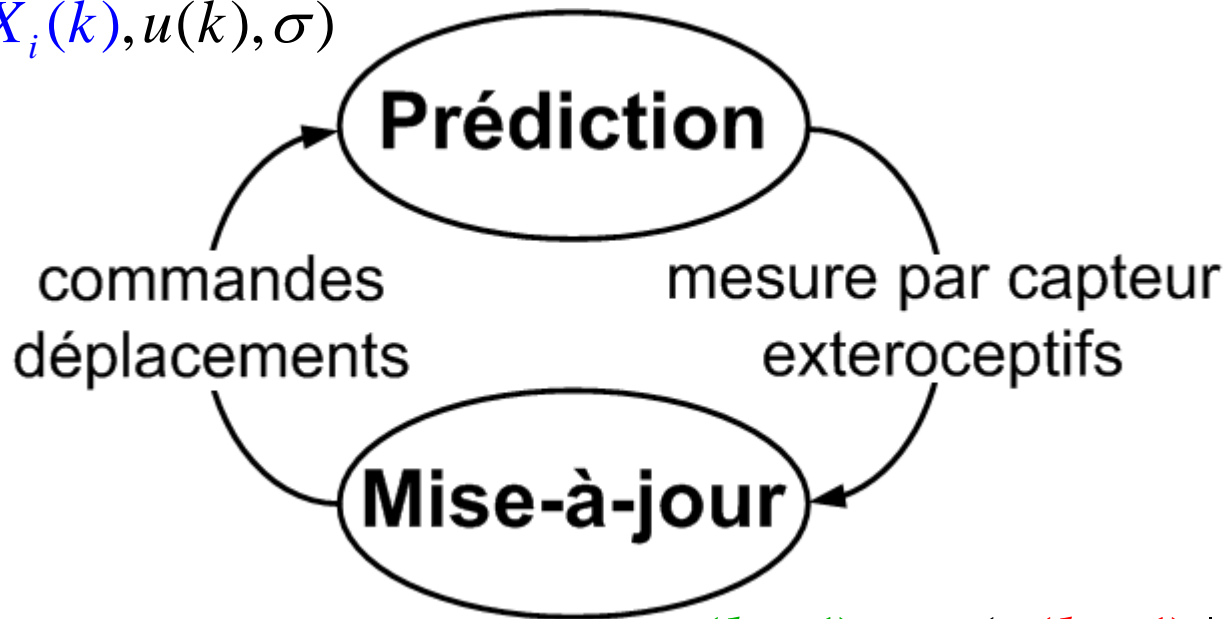
end while

Return(Index)

Prédiction + mise-à-jour

- Même mécanique de base que Kalman

$$X_i(k+1) = f_X(X_i(k), u(k), \sigma)$$



$$w_i(k+1) = p(z(k+1) | X_i(k+1))w_i(k)$$

Normaliser les w_i

Resampling, si $N_{eff} < N_{seuil}$

Filtre à particules

```
while (explore)
```

```
  for i=1:C
```

```
     $X_i(k+1) = f_X(X_i(k), u(k), \sigma)$       Prédiction
```

```
  end
```

```
   $z(k+1) = \text{mesure}()$  ;
```

```
  for i=1:C
```

```
     $w_i(k+1) = p(z(k+1) | X_i(k+1))w_i(k)$       Mise-à-jour
```

```
  end
```

```
  for i=1:C
```

```
     $w_i(k+1) = \frac{w_i(k+1)}{\sum_j \{w_j(k+1)\}}$       Normalisation
```

```
  end
```

```
  if ( $N_{eff} < N_{seuil}$ )
```

```
     $X_i(k+1) = \text{resample}(X_i(k+1), w_i(k+1))$  ;
```

```
     $w_i(k+1) = 1/C$ 
```

```
  end
```

```
  k=k+1
```

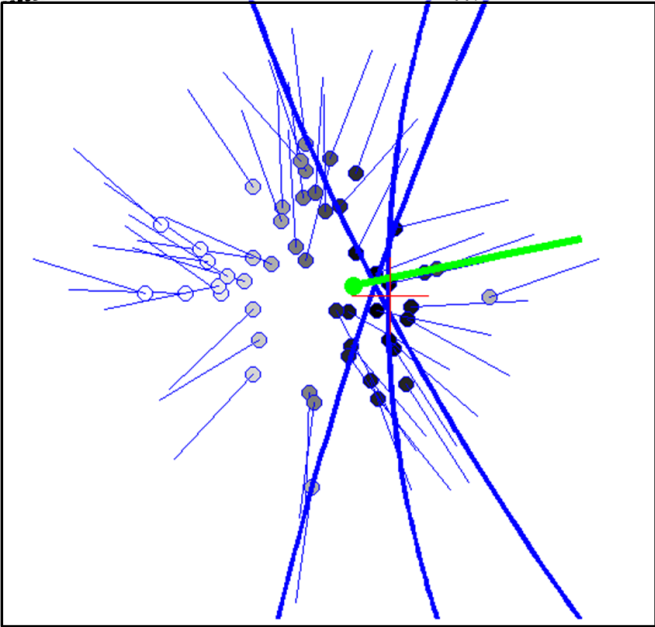
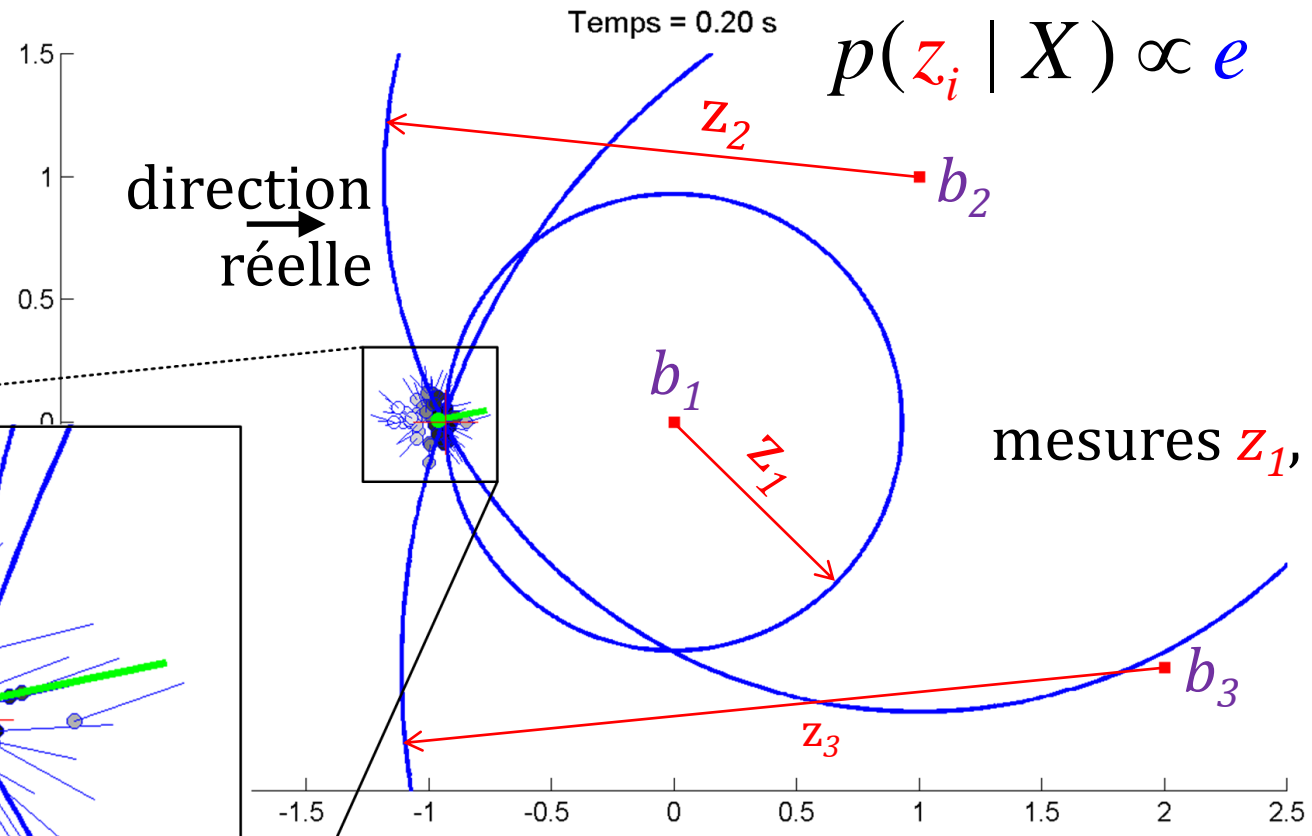
$$N_{eff} = \frac{1}{\sum_{i=1}^N w_i^2} < N_{seuil}$$

Ré-échantillonnage

Exemple 1 : particules + trilatération

$$\frac{(\|b_i - X\| - z_i)^2}{2\sigma^2}$$

- Moyenne pondérée position + angle
- Repères
- Poids w élevé
- Poids w moyen
- Poids w faible



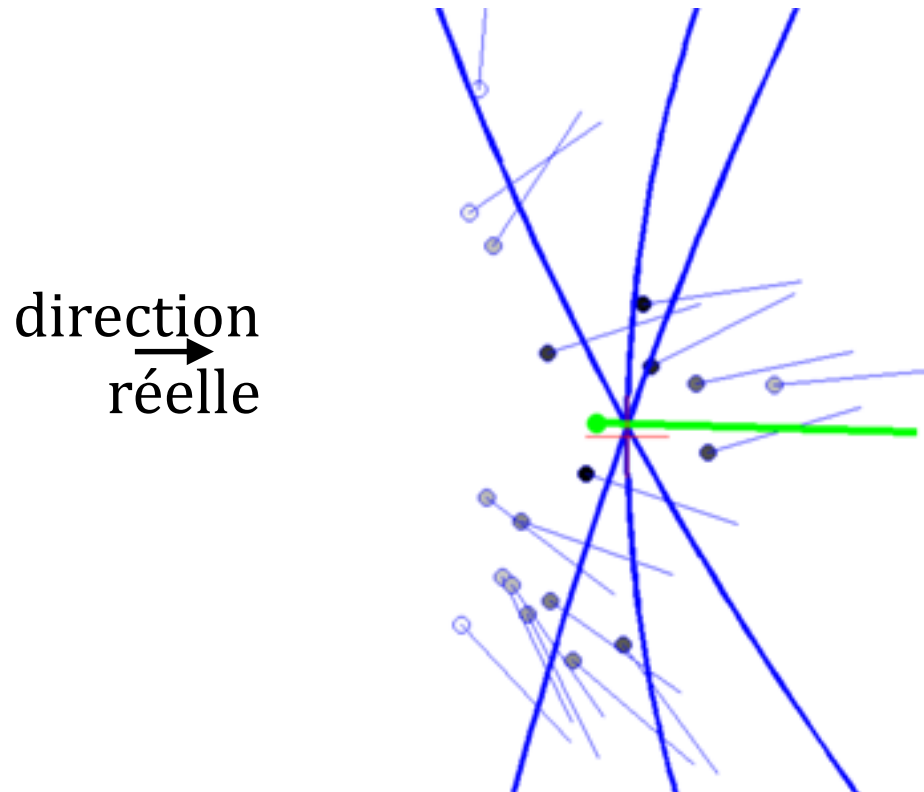
Après 1 déplacement

Initialisation de $C=50$ particules $X_i = [x \ y \ \theta]^T$, position initiale x, y connue, angles θ au hasard. Le système arrive à trouver un estimé de θ .



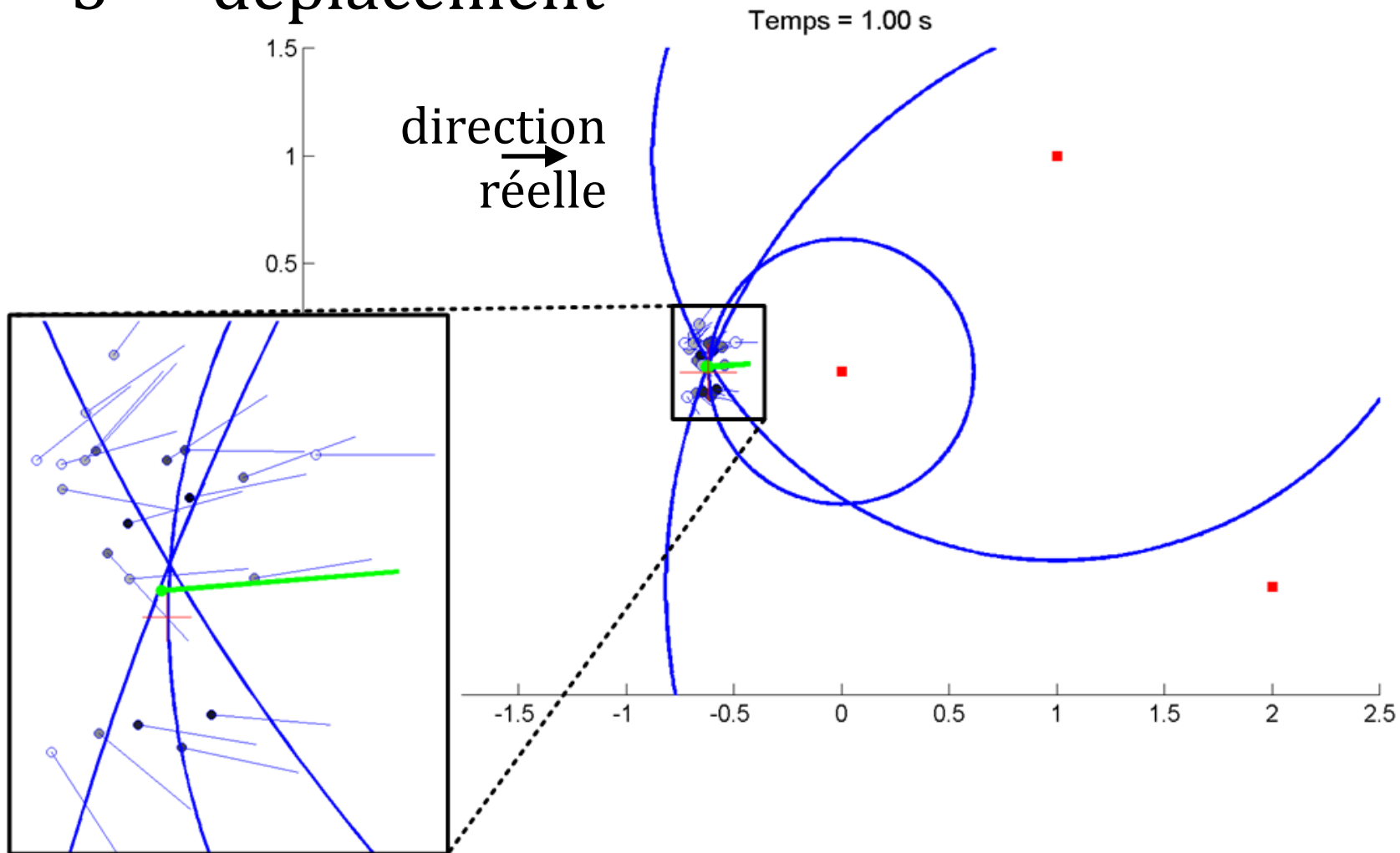
Exemple 1 : particules + trilatération

- Deuxième déplacement : presque toutes les particules avec le mauvais θ sont disparues



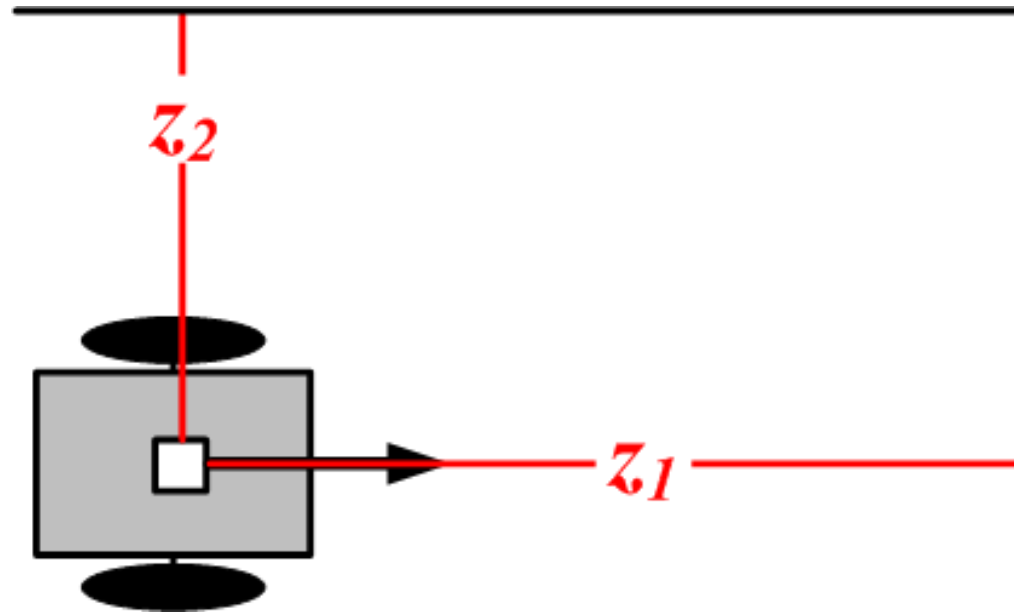
Exemple 1 : particules + trilatération

- 5^{ème} déplacement



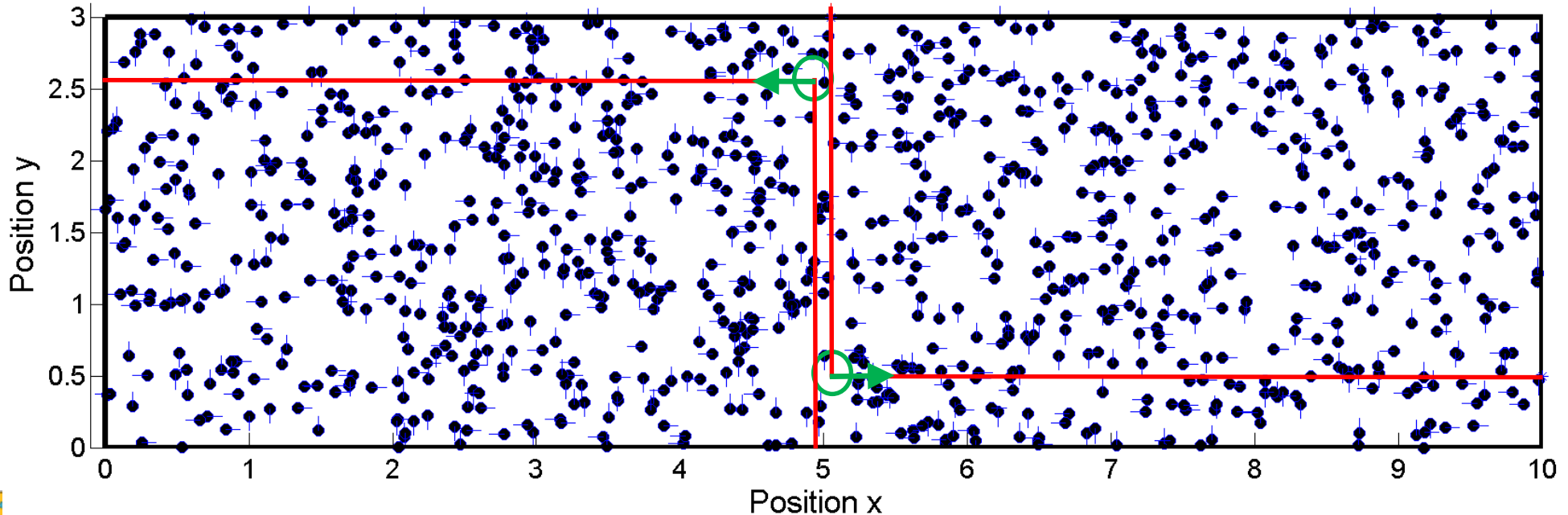
Exemple 2 : monde rectangle (similarité)

- Robot avec deux mesures laser : z_1 et z_2
- Pas de compas
- Carte connue



Exemple 2 : monde rectangle (similarité)

- Initialisé avec $C=400$ particules au hasard
 - position : uniformément (Problème de localisation globale)
 - angles : 0, 90, 180 ou 270 degrés
- Ligne rouge : mesures laser

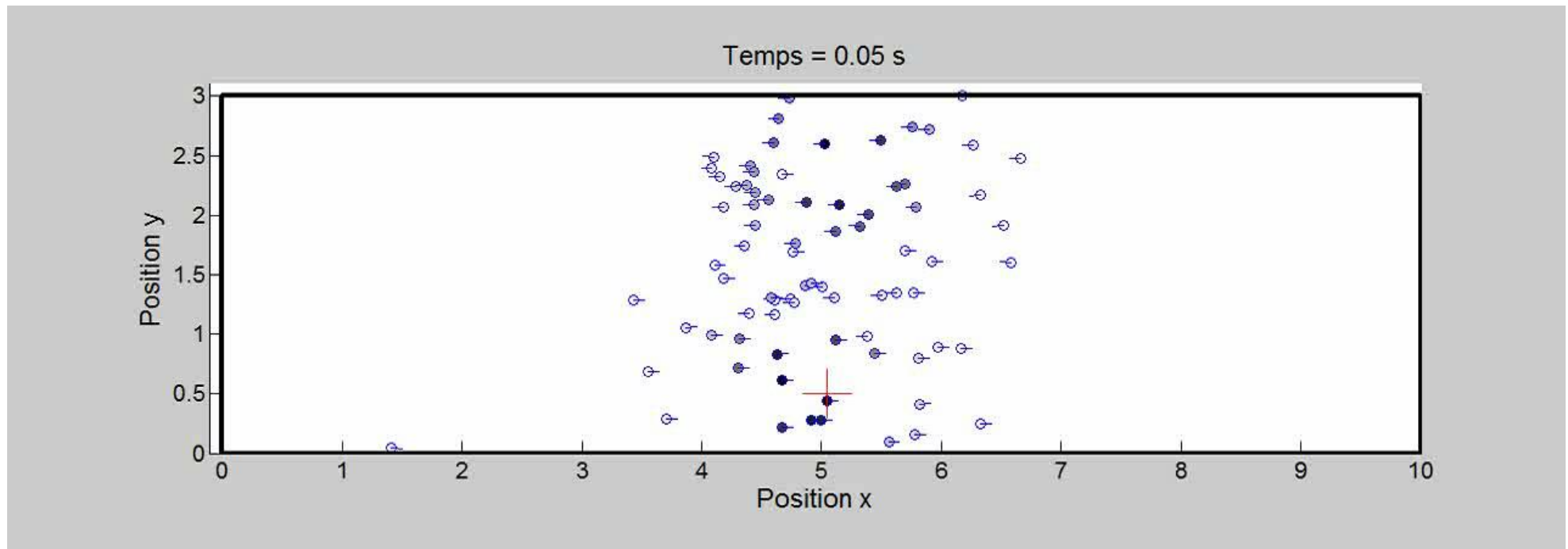


Exemple 2 : monde rectangle (similarité)

- Symétrie + absence de compas = ambiguïté

+ Position réelle du robot

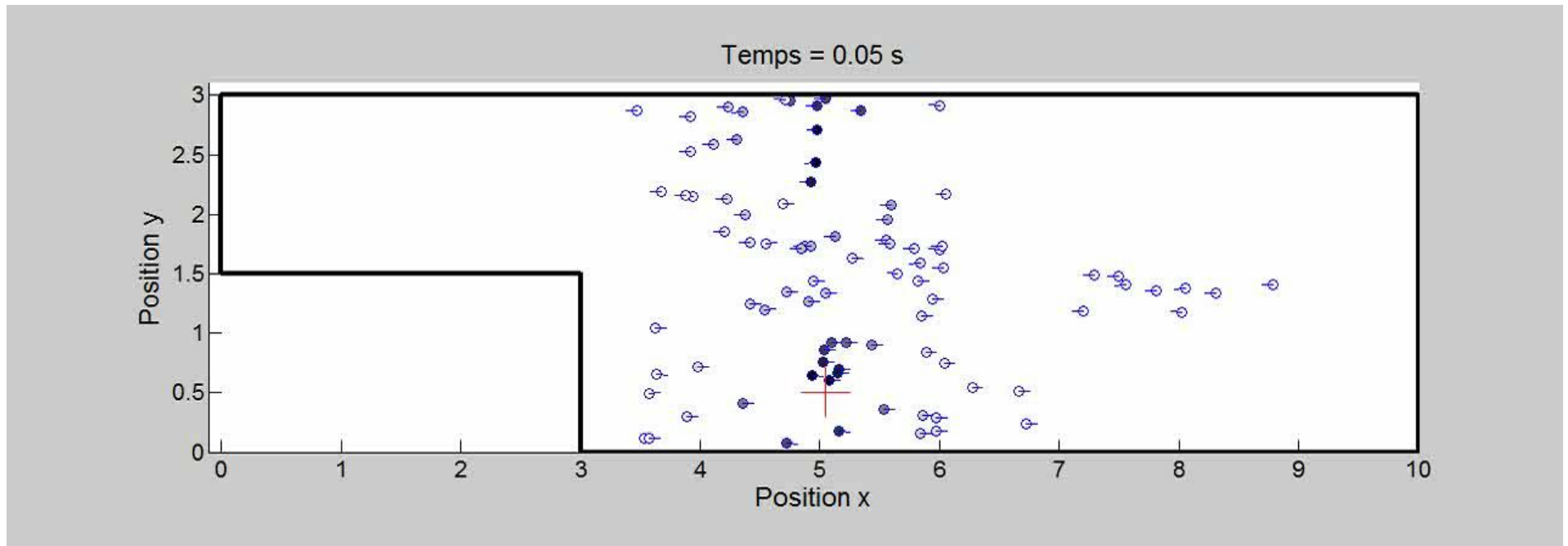
● Poids w élevé
● Poids w moyen
○ Poids w faible



Exemple 3 : monde asymétrique

- Asymétrie de la carte = levée de l'ambiguïté

● Poids w élevé
● Poids w moyen
○ Poids w faible

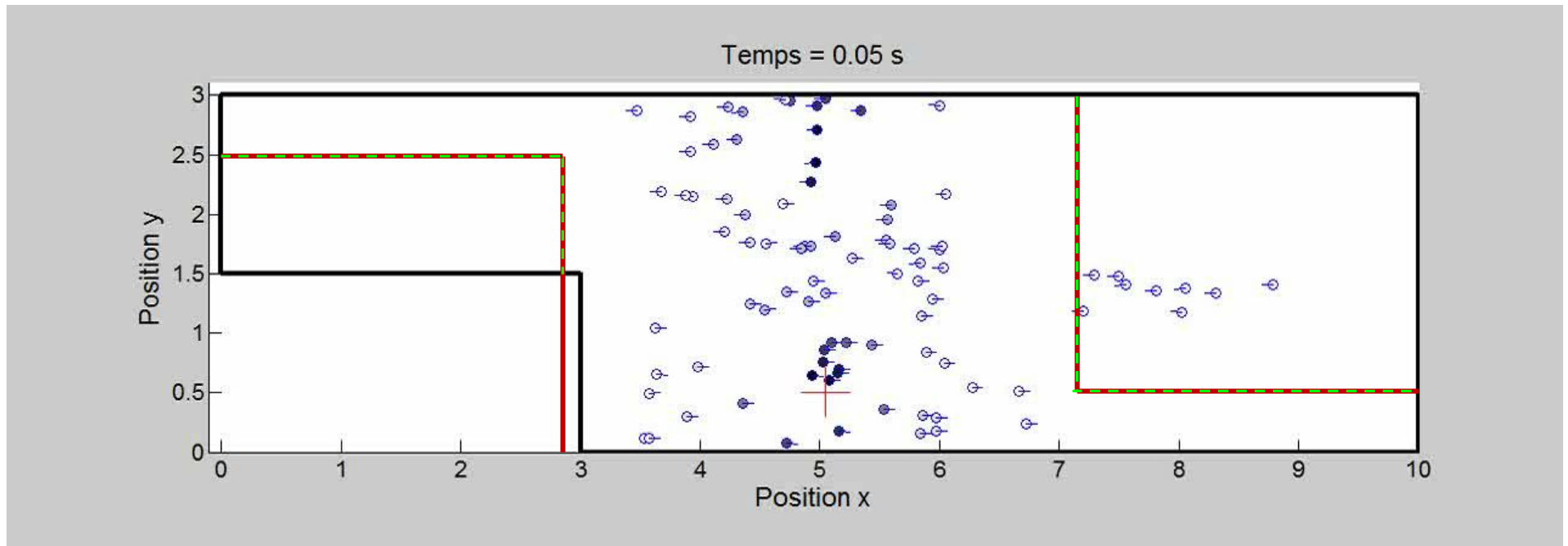


Exemple 3 : monde asymétrique

- Asymétrie de la carte = levée de l'ambiguïté

--- mesure estimée \hat{z}
— mesure réelle z

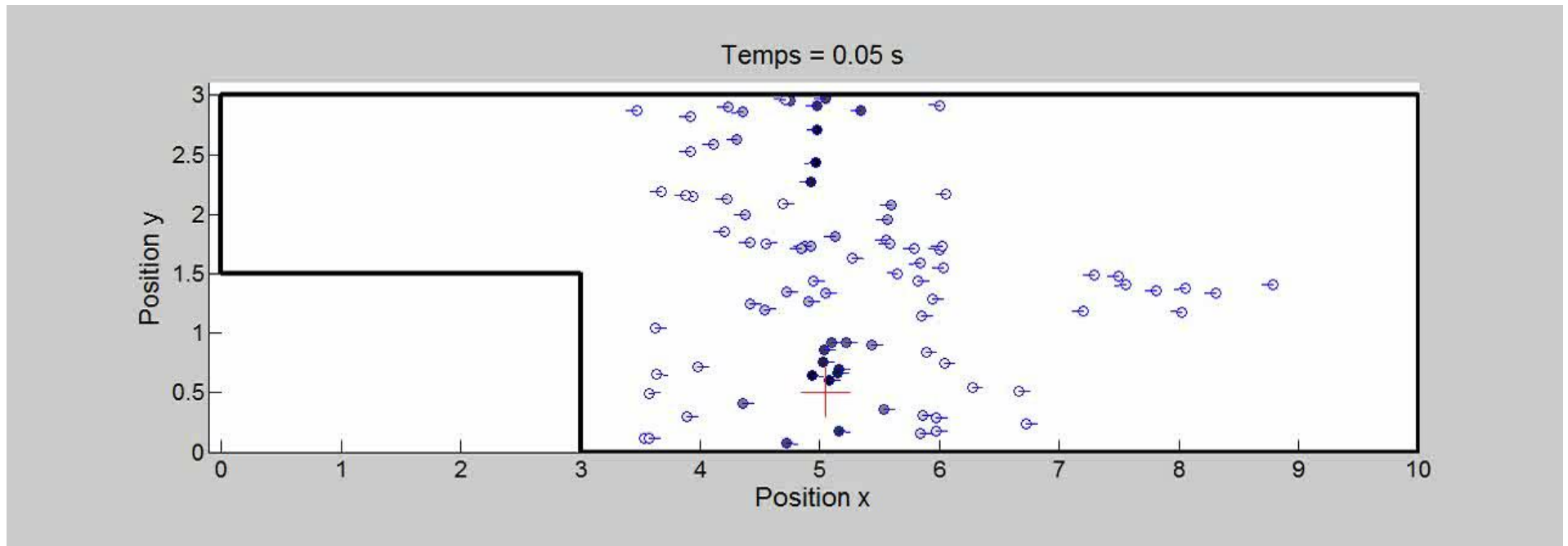
● Poids w élevé
● Poids w moyen
○ Poids w faible



Exemple 3 : monde asymétrique

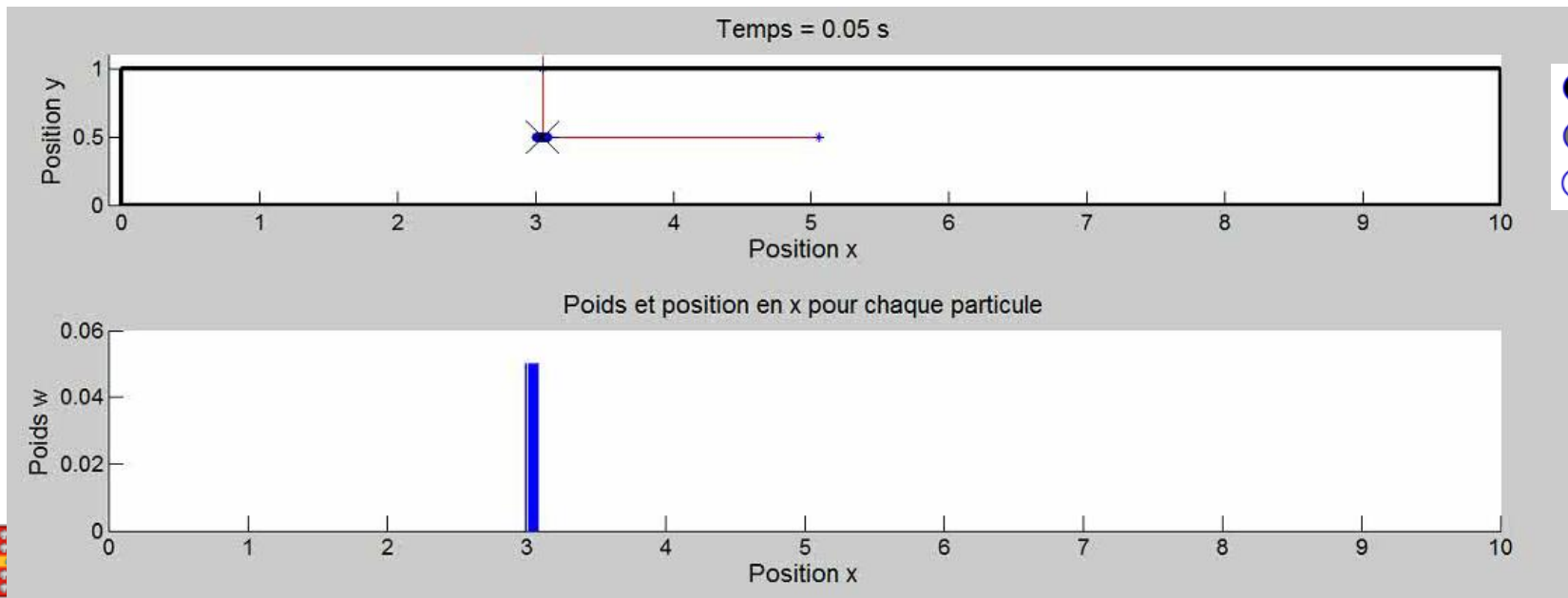
- Asymétrie de la carte = levée de l'ambiguïté

● Poids w élevé
● Poids w moyen
○ Poids w faible



Effet corridor

- Perte de précision le long de grands corridors
 - portée du LiDAR plus courte que la longueur du corridor
 - Exemple, portée LiDAR 2 m :
- ✗ Position réelle du robot



Bien adapté à la localisation globale

Quand on ne sait pas du tout où est la position de départ...



Trajectoire parcourue 

tiré de Probabilistic Robotics, par Thrun, Burgard et Fox

Bien adapté à la localisation globale



Trajectoire parcourue 

Bien adapté à la localisation globale



Trajectoire parcourue —

tiré de Probabilistic Robotics, par Thrun, Burgard et Fox

Localisation globale → locale

On détecte que le robot s'est localisé



On initialise un filtre de Kalman (étendu) : pistage (tracking)

Exemple 2: Fusion odom + gyro + GPS + carte!

- Bruit sur GPS : $\sigma = 50 \text{ m}$
- Bruit sur déplacement $\sigma = 2 \text{ m}$
- Odométrie + rotations du véhicule
- Heuristique : si la position n'est pas sur une route, réduire le poids de la particule de 90% :

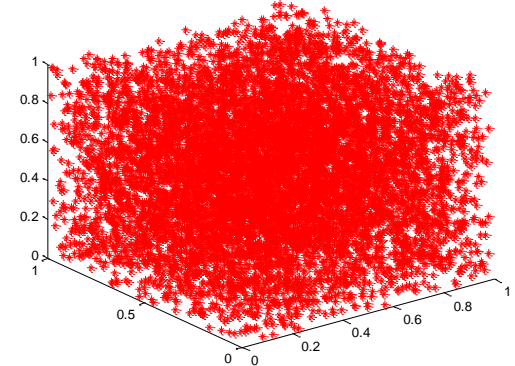
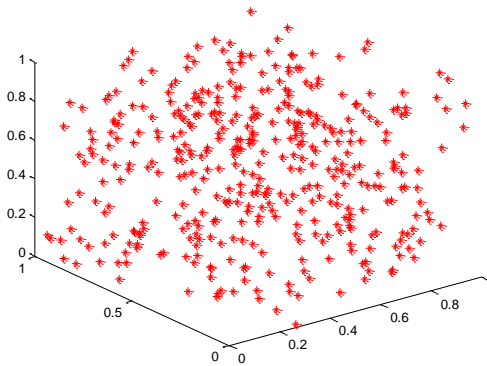
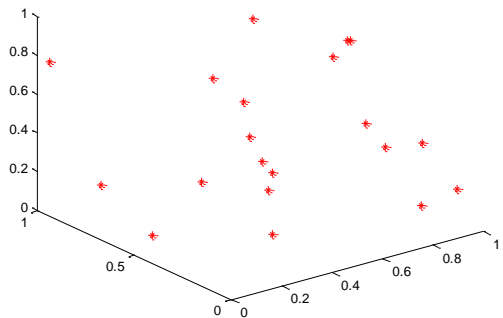
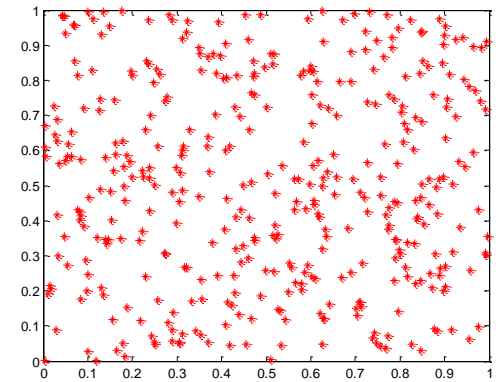
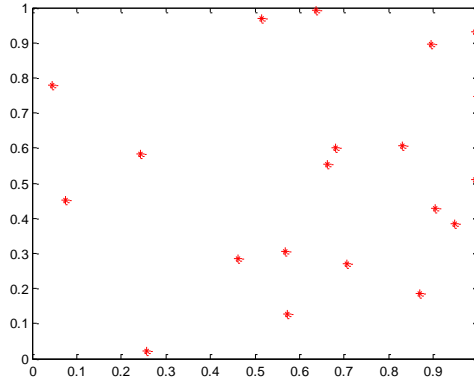
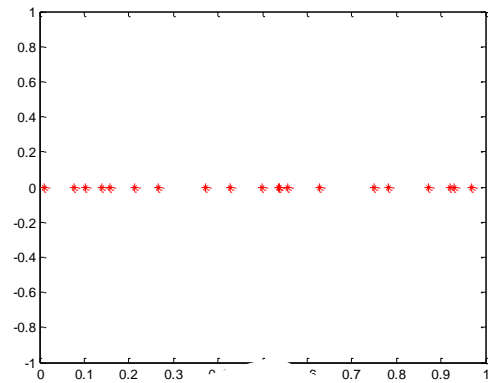
if (PasSurRoute($x(i)$))

$$w(i) = 0.1w(i)$$

end

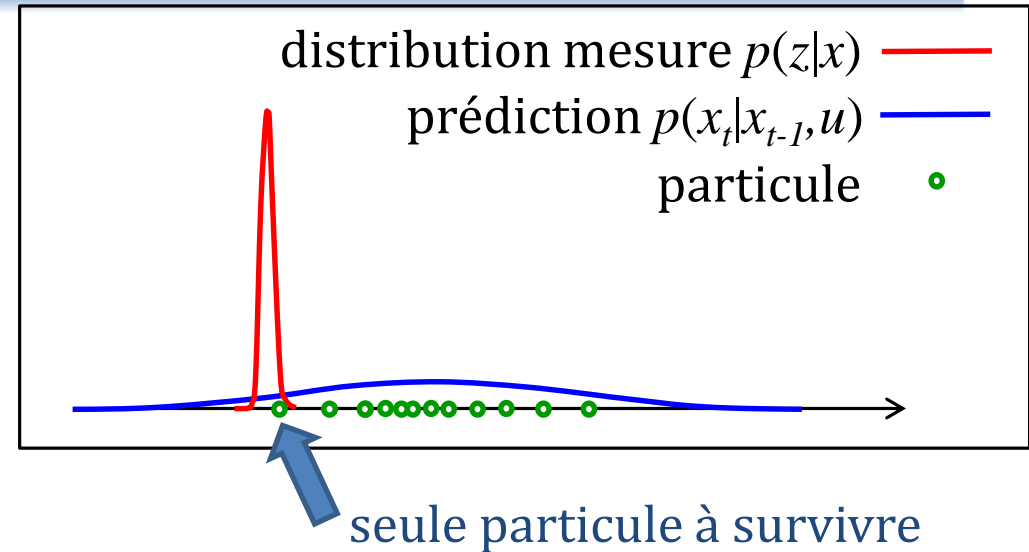
Gardez à l'esprit...

- Le nombre de particule C influence la précision
- Plus la dimensionnalité de l'état est grande, plus C doit être grand.



Problème si capteur trop précis...

- Peu de particules se trouvent dans la région couverte par la mesure
- Perte de variété (*deprivation*)
- Solutions:



- Placer des particules à l'endroit de la mesure

Lenser, Veloso. *Sensor resetting localization for poorly modelled mobile robots*. ICRA, 2000.

- Piger des particules dans une meilleure distribution combinant l'estimé de position et la mesure

- gmapping, FASTSLAM 2.0.

- Utiliser un bruit de capteur plus grand que la réalité

Variations pour filtres à particules

- Ajouter des particules uniformément après la mise-à-jour
 - *Augmente robustesse, survit au kidnapping*
- Ajouter des particules près de l'endroit indiqué par les capteurs
 - *prévient le problème de capteurs trop précis*
 - *difficile si l'inversion de la fonction de capteur ne nous donne pas une position*
- Augmenter ou diminuer le nombre C de particules en fonction du temps
 - *Adapting the sample size in particle filters through KLD-Sampling, D. Fox, IJRR, 2003.*

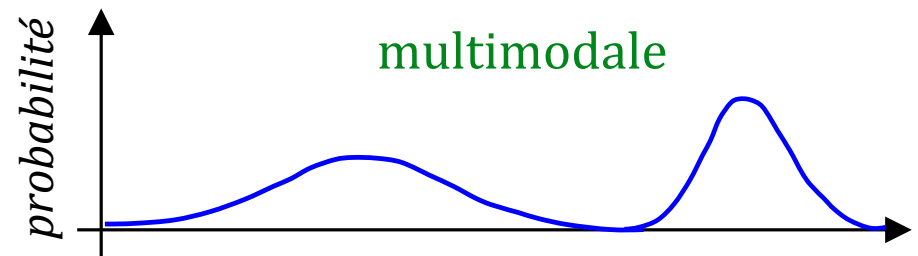
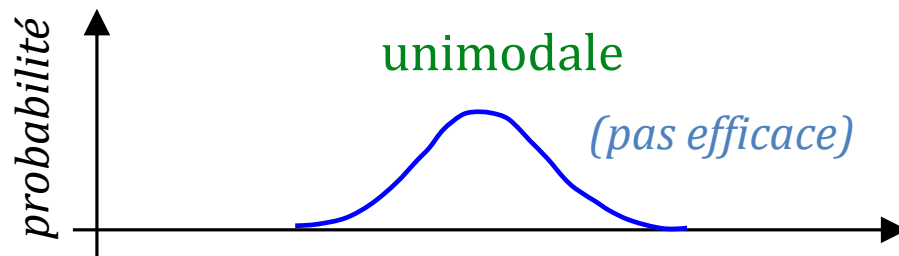
Rao Blackwellized Particle Filter (RBPF)

Filtre à particules Rao-Blackwellized

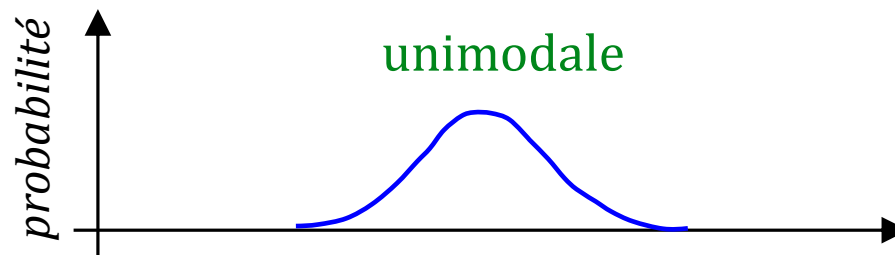
- **FP** : Si le nombre de variables d'état est grand, le nombre C de particules nécessaires est énorme
 - C est exponentiel en # variables
 - Fléau de la dimension
- **Filtre particule *Rao-Blackwellized*** :
 - Combiner FP + Kalman
 - Pour réduire le temps de calcul
 - Formulation exploite la structure présente dans le problème

Filtre à particules Rao-Blackwellized

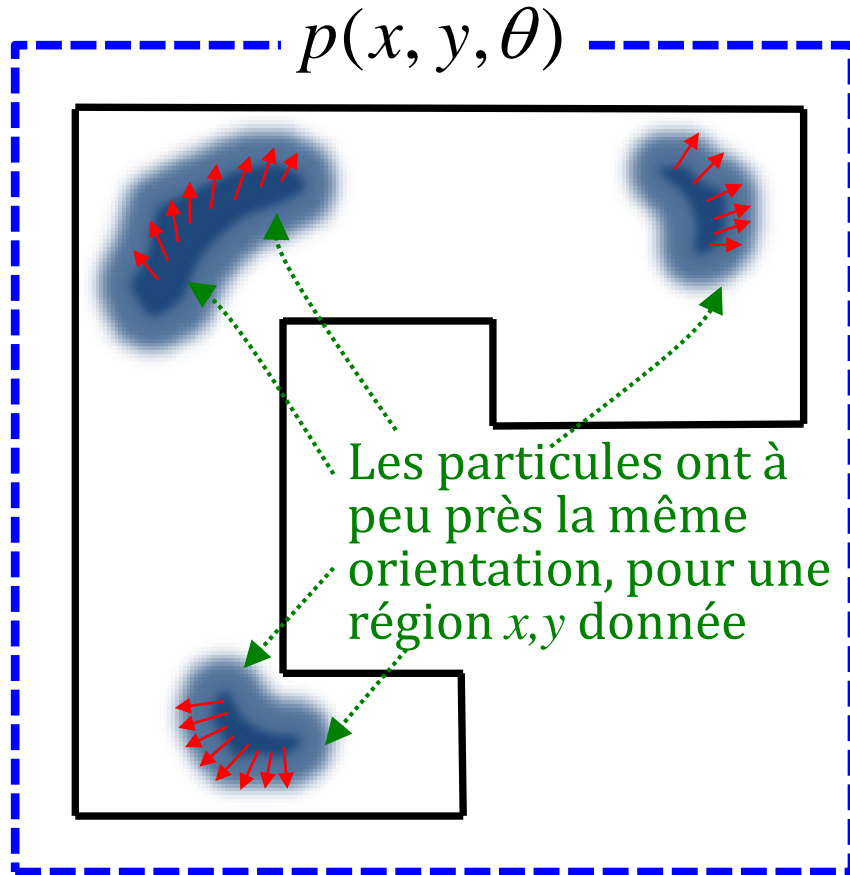
- **FP** : Peut représenter n'importe quel type de distribution (unimodale, multimodale)



- **Kalman** : la distribution à représenter doit être unimodale, idéalement gaussienne



Exemple RBPF pour localisation



Trois factorisations :

$$\begin{aligned} p(x, y, \theta) &= p(x | y, \theta) p(y, \theta) \\ &= p(y | x, \theta) p(x, \theta) \\ &\Rightarrow p(\theta | x, y) p(x, y) \end{aligned}$$

Y a-t-il une factorisation « meilleure » que les autres, en fonction de notre problème?

Si on fait l'hypothèse que sachant x, y , l'orientation θ est dist. unimodale, **oui!**

Factoriser en deux problèmes

Distribution
unimodale : Kalman

$$p(x, y, \theta) = p(\theta | x, y) p(x, y)$$

Distribution
multimodale : FP

Chaque particule utilisera donc
un filtre de Kalman!

(On y reviendra +tard pour FASTSLAM)

Bien visualiser le concept

- (1) $\hat{x}(k+1|k) = \Phi\hat{x}(k) + \Gamma u(k)$
- (2) $P(k+1|k) = \Phi P(k)\Phi^T + C_v$
- (3) $\hat{z}(k+1|k) = \Lambda\hat{x}(k+1|k)$
- (4) $r(k+1) = z(k+1) - \hat{z}(k+1|k)$
- (5) $K(k+1) = P(k+1|k)\Lambda^T \{\Lambda P(k+1|k)\Lambda^T + C_w(k+1)\}^{-1}$
- (6) $\hat{x}(k+1) = \hat{x}(k+1|k) + K(k+1)r(k+1)$
- (7) $P(k+1) = (I - K(k+1)\Lambda)P(k+1|k)$



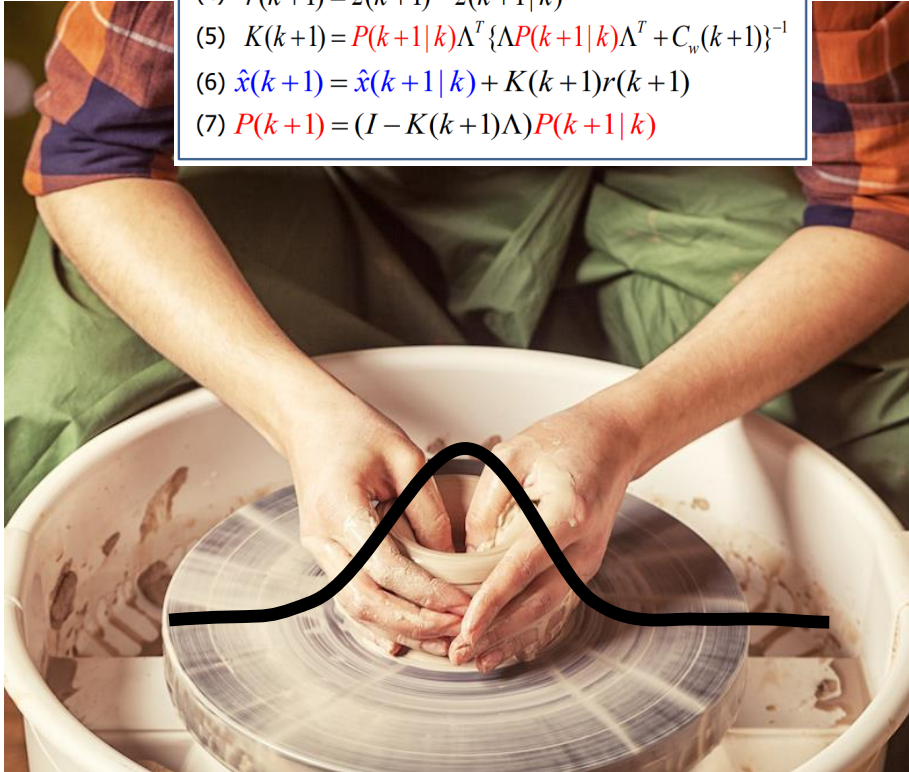
$p(x, y)$

Kalman

Filtre à particule

Bien visualiser le concept

- (1) $\hat{x}(k+1|k) = \Phi\hat{x}(k) + \Gamma u(k)$
- (2) $P(k+1|k) = \Phi P(k)\Phi^T + C_v$
- (3) $\hat{z}(k+1|k) = \Lambda\hat{x}(k+1|k)$
- (4) $r(k+1) = z(k+1) - \hat{z}(k+1|k)$
- (5) $K(k+1) = P(k+1|k)\Lambda^T \{\Lambda P(k+1|k)\Lambda^T + C_w(k+1)\}^{-1}$
- (6) $\hat{x}(k+1) = \hat{x}(k+1|k) + K(k+1)r(k+1)$
- (7) $P(k+1) = (I - K(k+1)\Lambda)P(k+1|k)$



$p(x, y)$

Kalman

Filtre à particule

Commandes et mesures désynchronisées

Filtres : mesures à différents rythmes?

- Commandes aux 100 ms
- Capteur 1 aux 300 ms
- Capteur 2 aux 500 ms

Autant pour les filtres à particules
que les filtres de Kalman

Propage commande
Mise-à-jour mesure z1
Propage commande
Propage commande
Propage commande
Mise-à-jour mesure z1
Propage commande
Propage commande
Mise-à-jour mesure z2
Propage commande
Mise-à-jour mesure z1
Propage commande
Propage commande
Propage commande
Mise-à-jour mesure z1
Propage commande
Mise-à-jour mesure z2
Propage commande
Propage commande
Mise-à-jour mesure z1

Types de problème de localisation

Problèmes de localisation

- **Localisation faible**


- « Ai-je déjà été ici ? »
- marqueurs déposés au sol
- algorithmes de planification bug_0 , bug_1 , bug_2
- reconnaissance de lieu (*place recognition*)



- **Localisation forte**

- « Où suis-je par rapport à un repère global ? »
- $p(x_t/z_{1:t}u_{1:t})$
 - x_t : pose actuelle du robot
 - $z_{1:t}$: mesures des capteurs et $u_{1:t}$: commandes

Localisation forte

- **Pistage/Tracking (unimodale)**
 - je connais ma position initiale x_0
 - implémenté par filtre Kalman
- **Globale (multimodale)**
 - je ne connais pas ma position initiale x_0
 - implémenté par filtre à particule, etc.
- **Kidnapping**
 - téléportation possible du robot sans l'avertir
 - détecte algorithme qui diverge/échoue
 - globale \rightarrow tracking  globale \rightarrow tracking

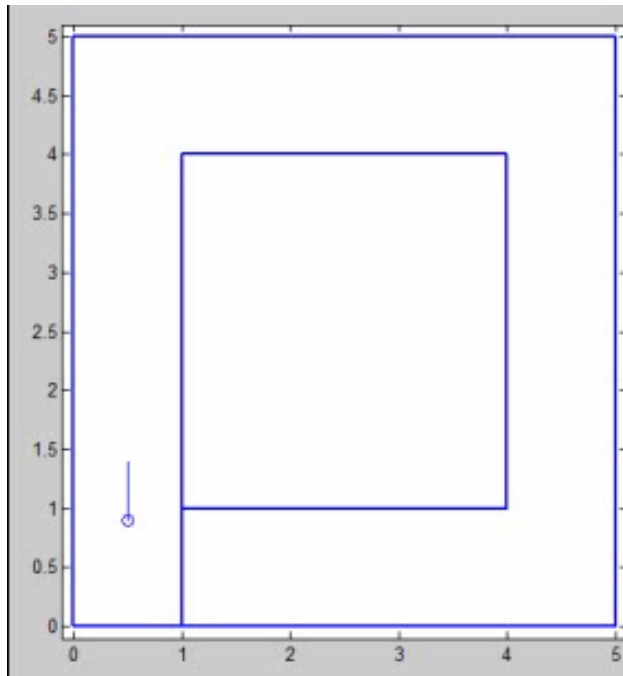
difficulté croissante

Iterative Closest Point : ICP

Positionnement relatif par scan 2D

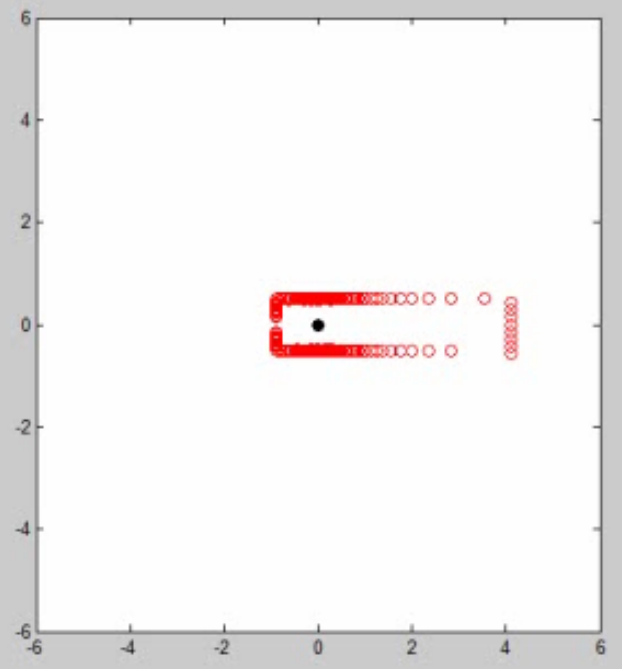
- Utiliser directement les données Z d'un capteur de distance pour estimer la position relative entre deux scans.
- Scénario :

**Monde m +
position du robot**



coordonnées globales

Scan laser

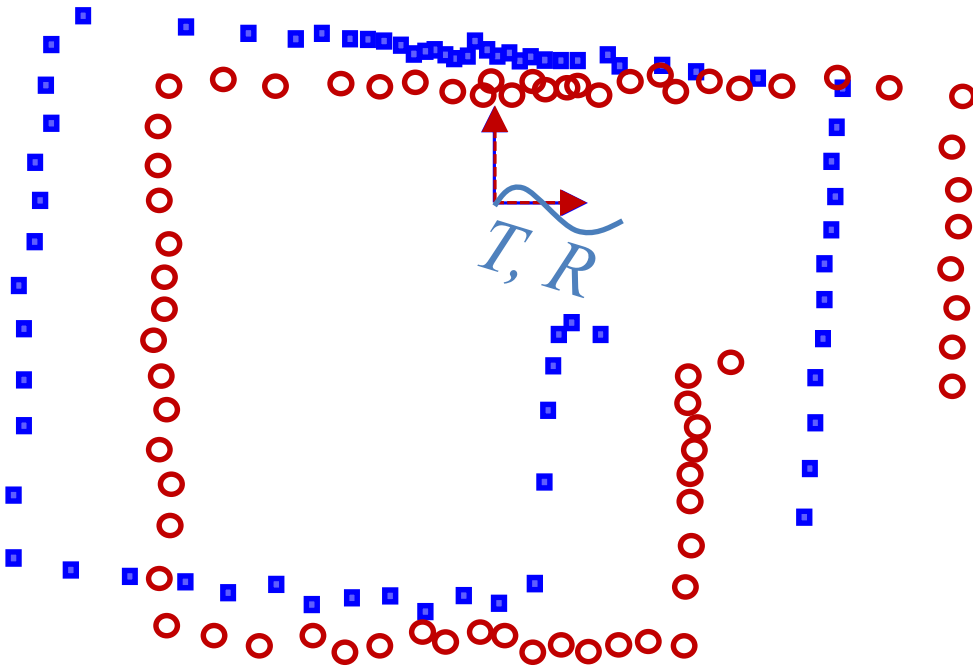


coordonnées locales (robot)



Opération de calage de points

- Deux scans lidar 2D/3D, A et B , pris à des endroits différents mais proches
- Problème : trouver R et T entre A et B

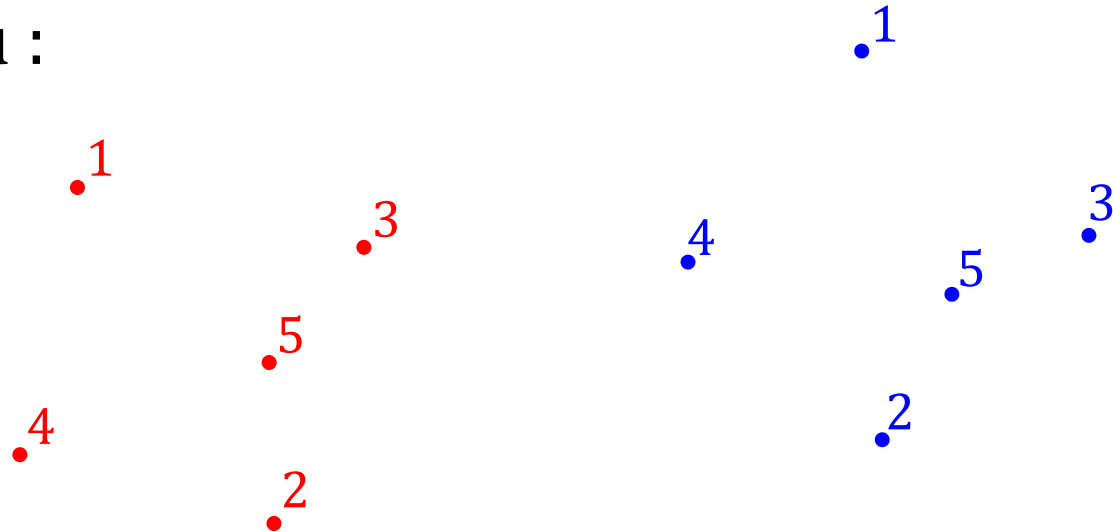


- Similaire à odométrie visuelle
- **Sauf que les points lidar sont anonymes ...**
- **et les mesures seront rarement pour une même partie de surface**



Problème de calage de point (simple)

- Ici on assume que l'appariement entre points **A** et **B** est exact et connu :

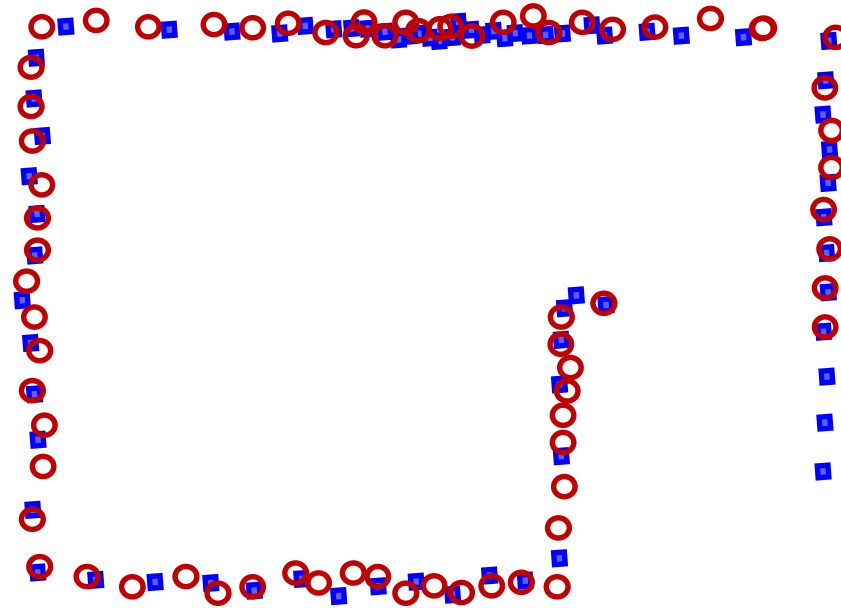


- Cherche : $\operatorname{argmin}_{R,T} \sum_i \| A_i - (T + RB_i) \|^2$

- Se résout en 10 lignes matlab, sans itération (SVD)

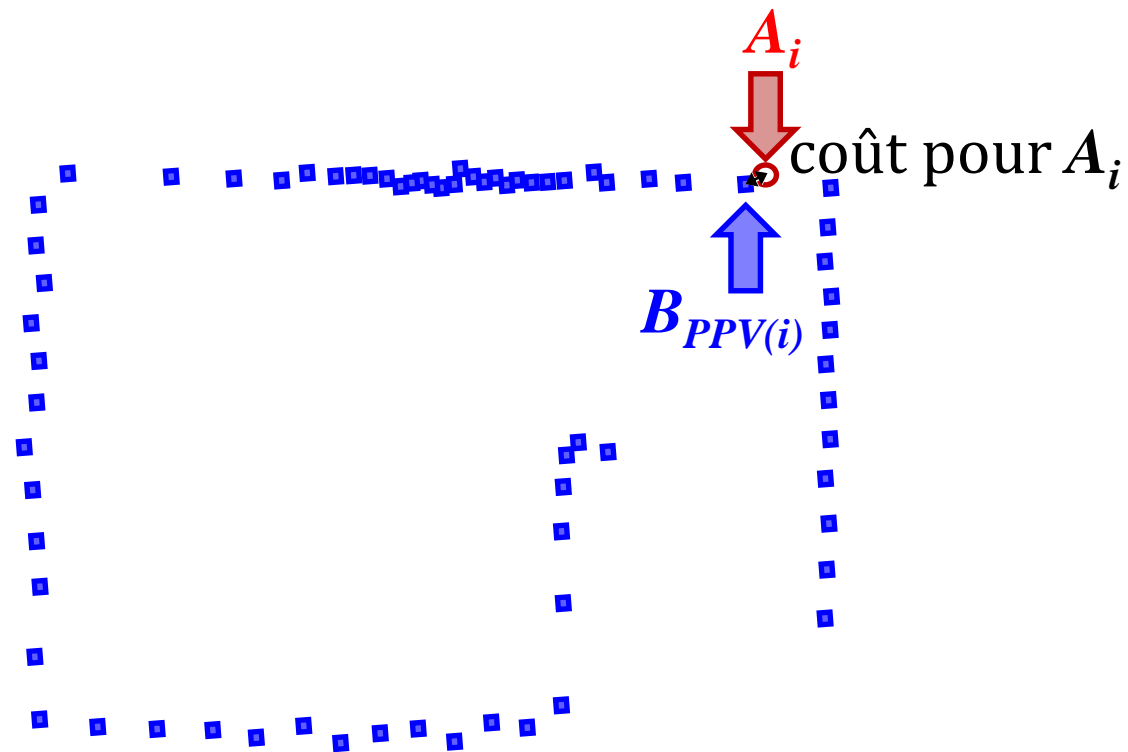
Plus proche voisin pour calcul coût

- Les points n'ont pas de signatures
- Apparier points par critère du plus proche voisin $PPV(i)$.



$$\operatorname{argmin}_{R,T} \sum_{i \in A} \| A_i - TRB_{PPV(i)} \|^2$$

Plus proche voisin pour calcul coût



$$\operatorname{argmin}_{R,T} \sum_{i \in A} \| A_i - TRB_{PPV(i)} \|^2$$

ICP : Iterative Closest Point

- Fonctionne par ajustement itératif avec T et R
- Boucler jusqu'au critère de convergence :
 - Faire l'appariement $PPV(i)$
 - Estimer T et R qui minimisent* l'erreur :
$$A_i \quad B_{PPV(i)}$$
 - Applique la transformation : $B \leftarrow T + RB$
- **Attention!** ne fonctionne que pour des petites transformations T et R .
 - sorte de descente de gradient
 - sujet aux minimums locaux, si grands déplacements T et R

**solution approximative car dépend appariement, d'où le besoin de faire de multiples itérations*

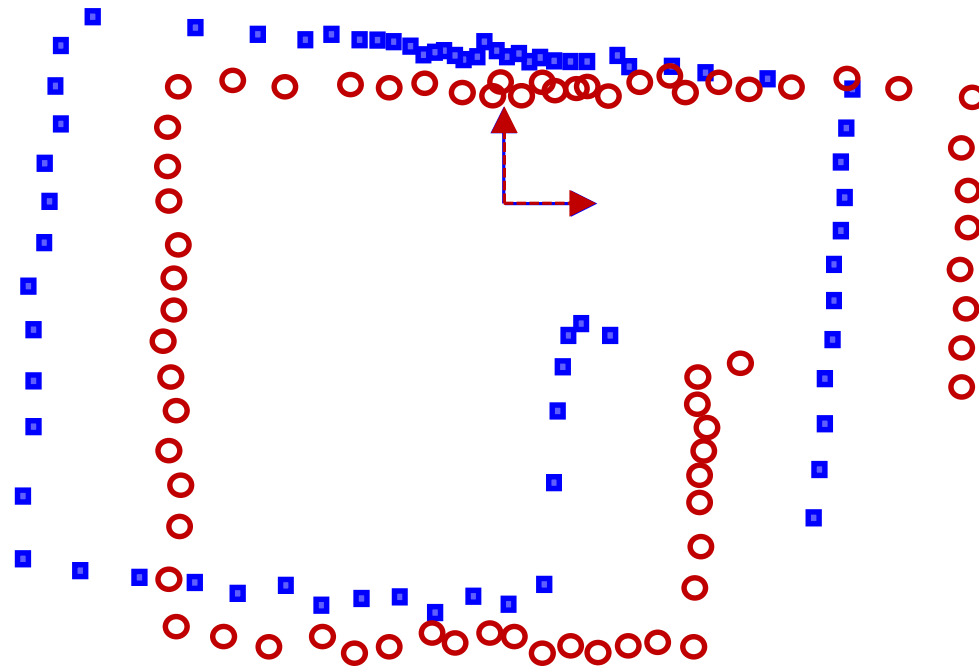
ICP : Iterative Closest Point

- Fonctionne par ajustement itératif avec T et R
- Boucler jusqu'au critère de convergence :
 - Faire l'appariement $PPV(i)$
 - Estimer T et R qui minimisent* l'erreur :
$$\operatorname{argmin}_{R,T} \sum_{i \in A} \| A_i - (T + RB_{PPV(i)}) \|^2$$
 - Applique la transformation : $B \leftarrow T + RB$
- **Attention!** ne fonctionne que pour des petites transformations T et R .
 - sorte de descente de gradient
 - sujet aux minimums locaux, si grands déplacements T et R

**solution approximative car dépend appariement, d'où le besoin de faire de multiples itérations*

Position de départ de recherche ICP

Odométrie + gyroscope pour trouver une position initiale

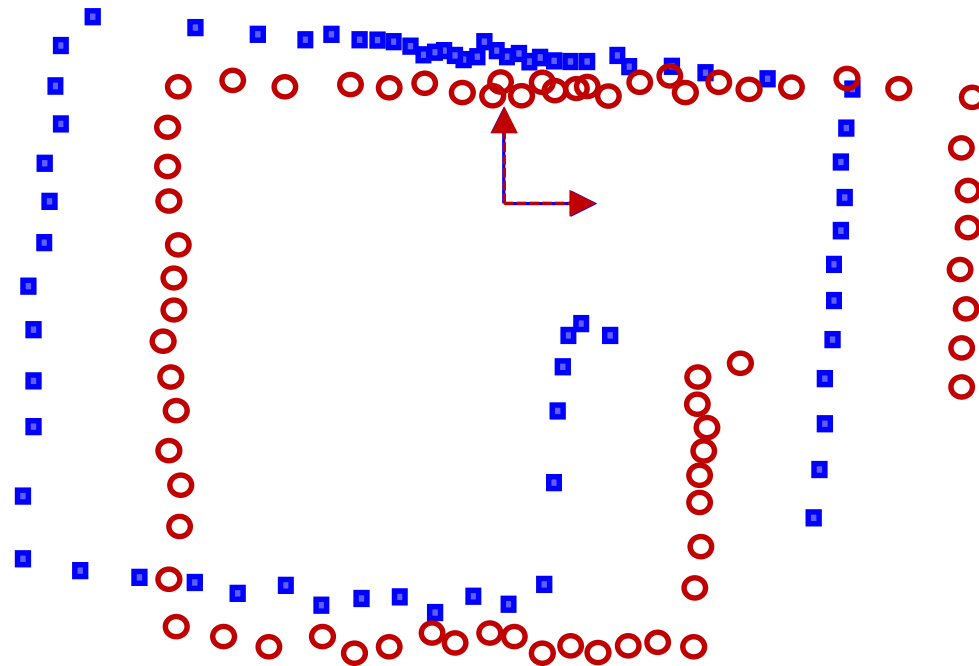


Proprioceptif

Avant ajustement odométrie

Position de départ de recherche ICP

Odométrie + gyroscope pour trouver une position initiale

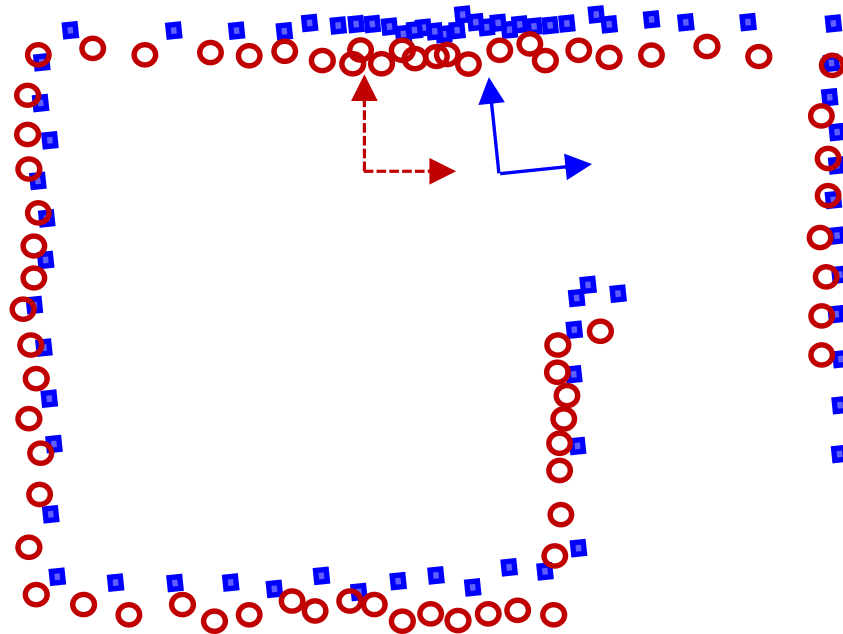


Proprioceptif

Avec ajustement odométrie

Position de départ de recherche ICP

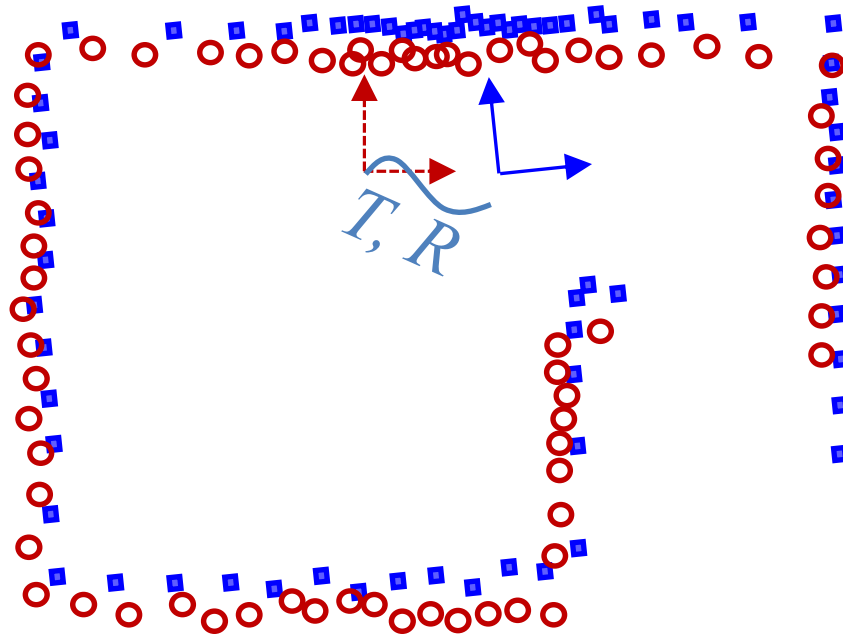
- Applique ensuite ICP, car on est proche du minimum local



Extéroceptif

Position de départ de recherche ICP

- Applique ensuite ICP, car on est proche du minimum global : raffine la position



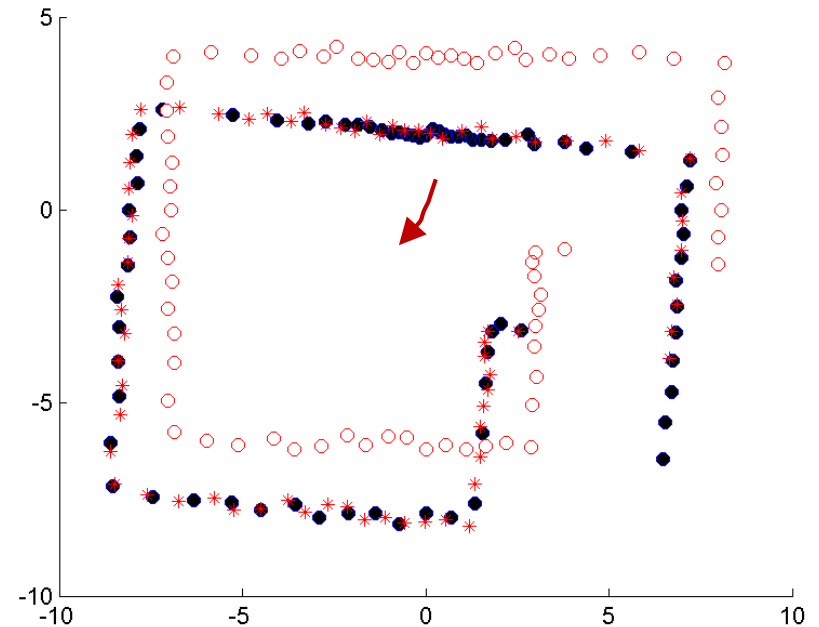
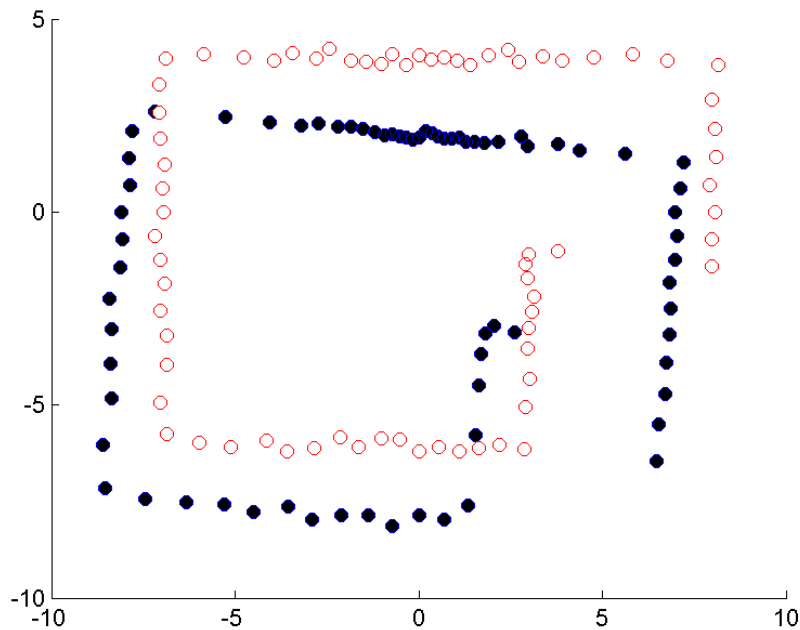
Extéroceptif

Exemple ICP

- Fonction `icp.m` de Per Bergström

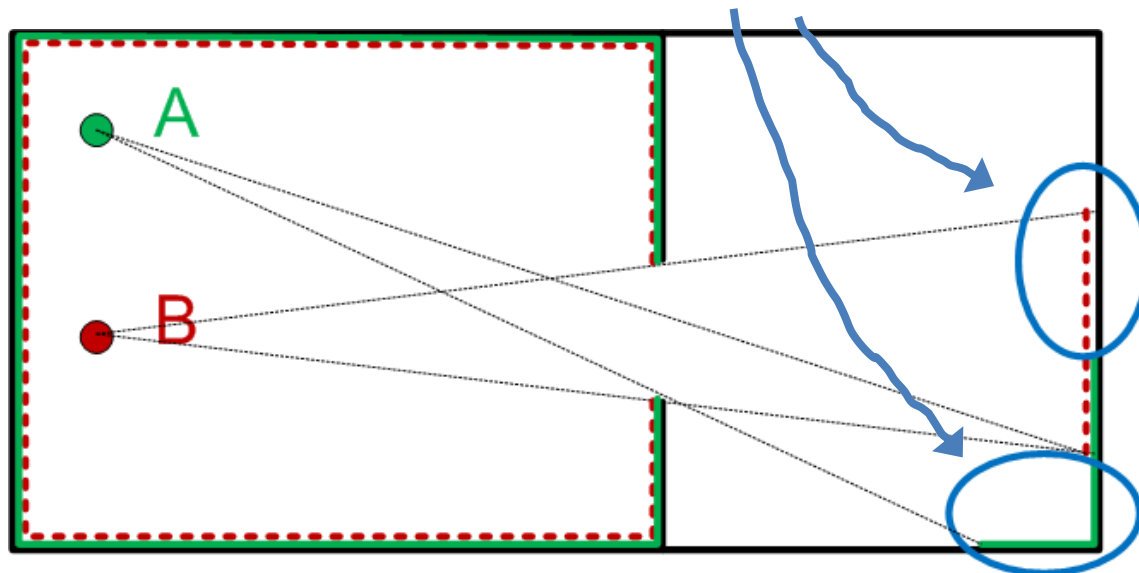
(<http://www.mathworks.com/matlabcentral/fileexchange/12627>)

Laboratoire de ce vendredi



ICP : Problème des données aberrantes

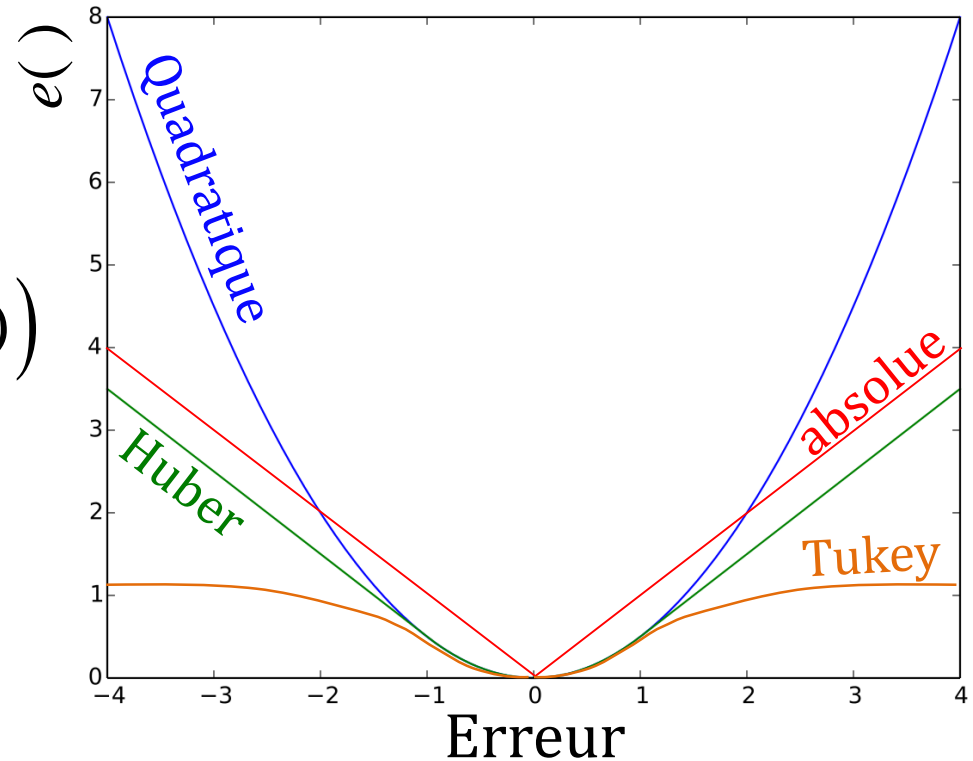
- Quant il y a des occlusions dans la scène
 - pièce n'est pas convexe
- Certains points ne peuvent pas être apparié
- Présence de données aberrantes (*outliers*).



Données aberrantes : fonctions robustes

- Utiliser d'autres fonctions d'erreurs

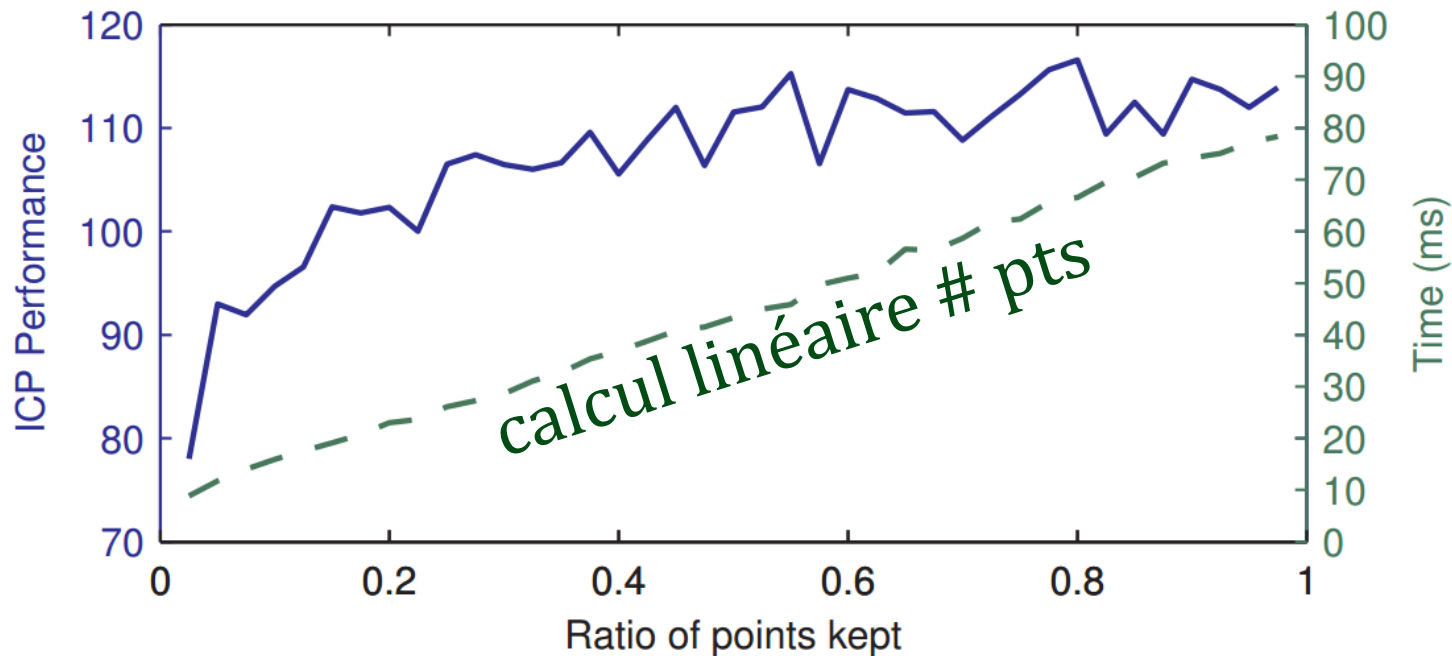
$$\operatorname{argmin}_{R,T} \sum_{i \in A} e\left(A_i - (T + RB_{PPV(i)})\right)$$



- Retirer % des erreurs les plus grandes

Considération : # points

- Possible d'accélérer les performances en décimant le nombre de points



Tracking a Depth Camera: Parameter Exploration for Fast ICP,
François Pomerleau et al., *IROS* 2011.