ENSTA

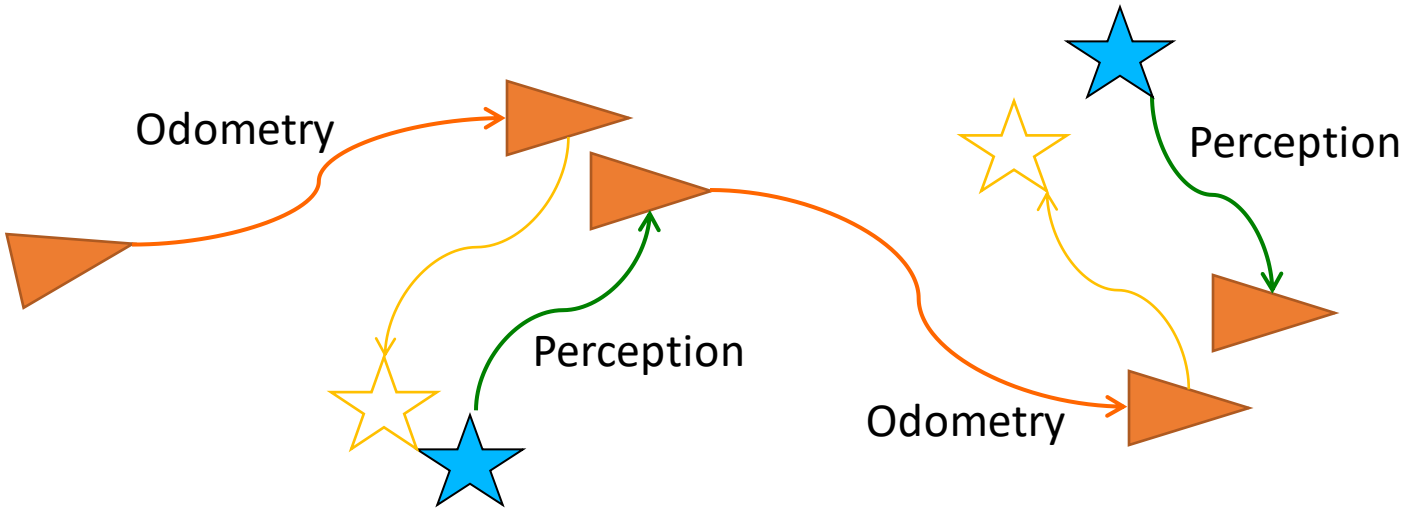
IP PARIS

ROB201 – Introduction à la robotique mobile

Cours 04 – Localisation, SLAM – David Filliat / Thibault Toralba

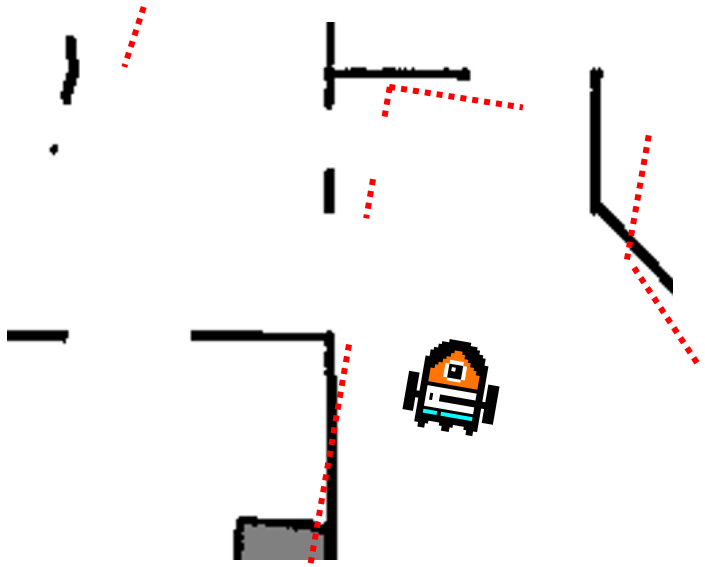
Localisation

- Rappel : Données bruités
  - Besoin de résoudre les ambiguïtés de perception
  - Besoin de corriger la dérive de l'odométrie
  - Fusion des deux types d'information

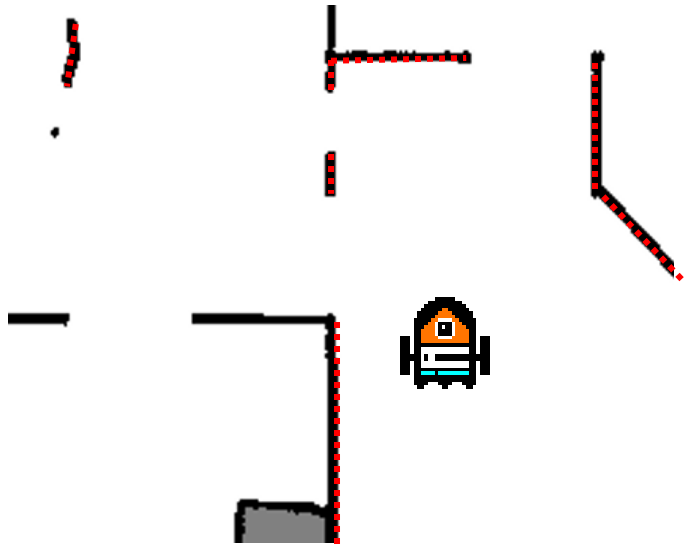


Localisation

- Utilisation de la carte pour associer odométrie / perception
- **Prédiction** de position par l'odométrie
- **Correction** par alignement du scan sur la carte



Position prédite par odométrie



Position corrigée par perceptions

Localisation : ICP (voir 3A)

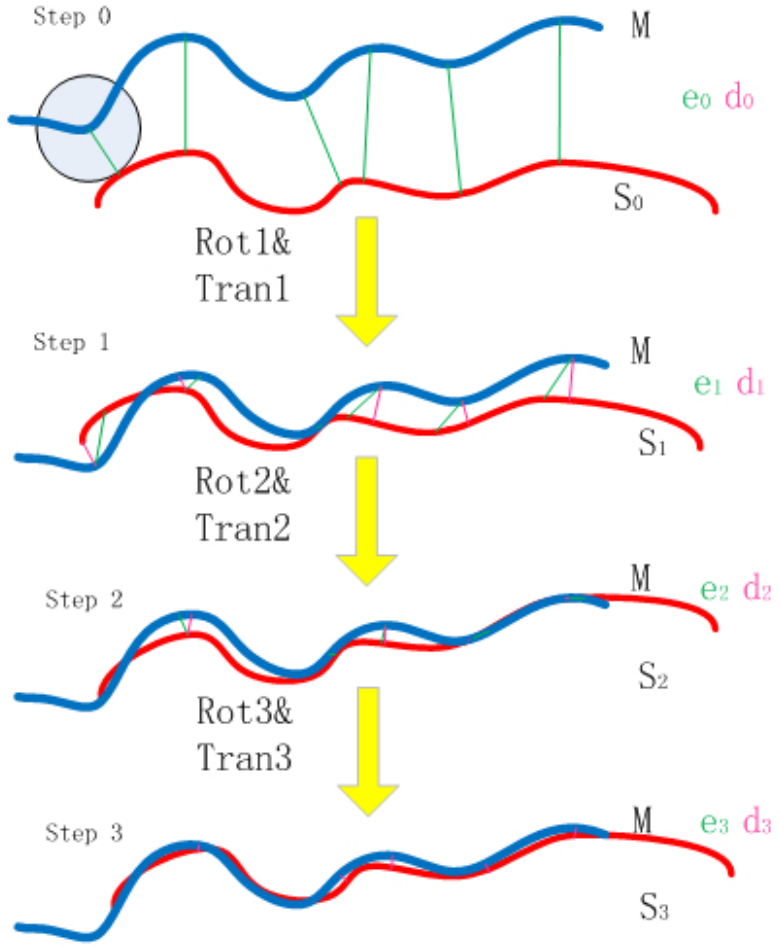
- Différentes méthodes en fonction des capteurs et des cartes
- Ex : Iterated Closest Point

  Capteurs et carte sous forme de nuage de points

  Calculer la transformation superposant des scans est facile si on connait les correspondances des points, mais elle est inconnue → heuristique réitérée

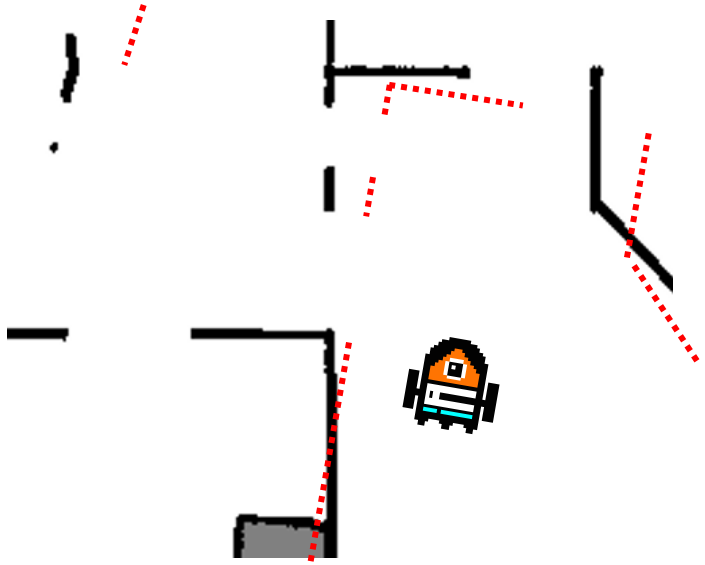
  - Initialiser les points p_i d'une position estimée (odométrie par ex.)
  - Répéter
    - Associer chaque point du scan à la référence
    - Calculer R,t en minimisant $f(R,t)=\frac{1}{n}\sum_{i=1}^{n}[\vec{p}_i-(R\vec{p}'_i+t)]^2$
    - Appliquer R,t au scan

  Tant que f(R,t) > seuil ET nbr_iter < iter_max

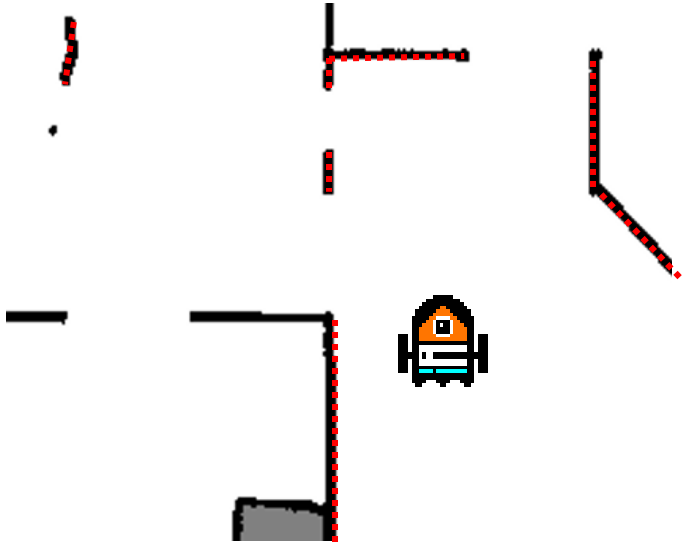


Localisation : Fonction de score

- Recherche de la position optimisant l'alignement du nuage de points laser avec les obstacles de la grille d'occupation
- Fonction de score: somme des probabilités des cellules ayant un impact laser



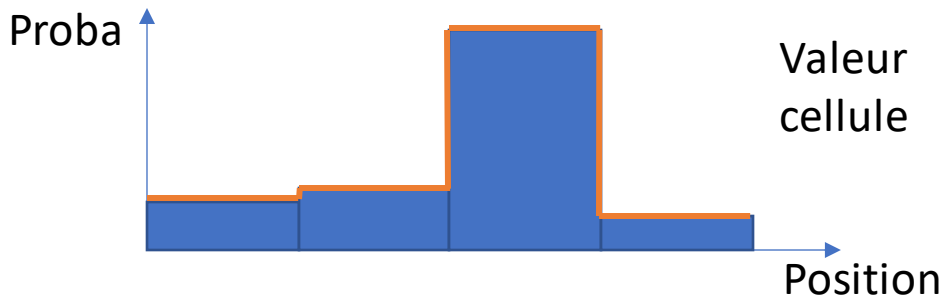
Score = 7,34

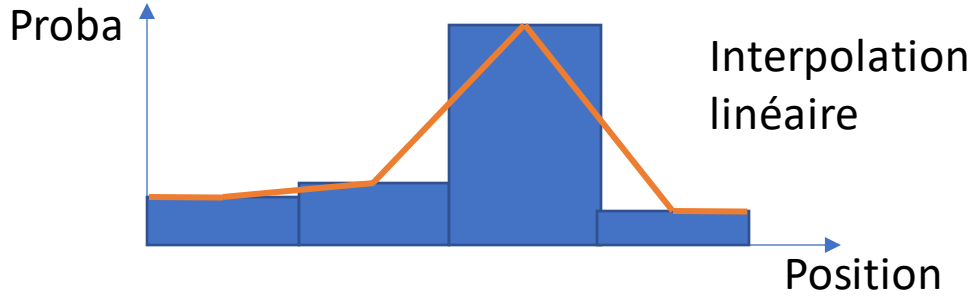


Score = 357,2

Localisation : Fonction de score

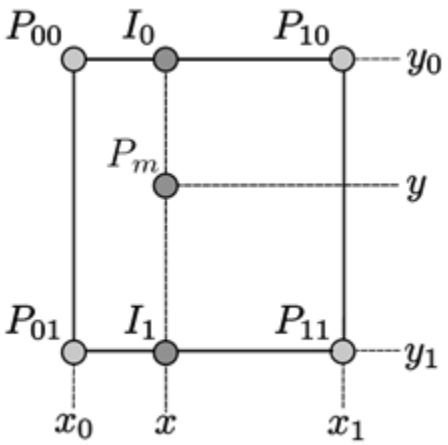
- Fonction de score plus précise : interpolation bilinéaire
- En 1D, interpolation linéaire donne des maximums mieux définis





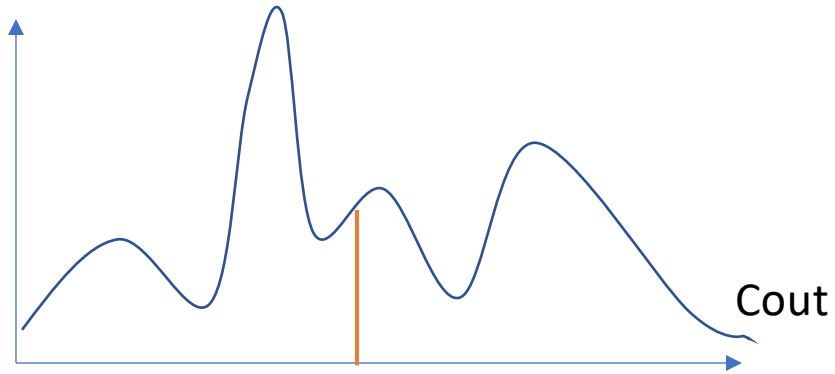
- En 2D, implémentable ‘rapidement’ avec numpy

$$M(P_m) \approx \frac{y - y_0}{y_1 - y_0}\left(\frac{x - x_0}{x_1 - x_0}M(P_{11}) + \frac{x_1 - x}{x_1 - x_0}M(P_{01})\right) + \frac{y_1 - y}{y_1 - y_0}\left(\frac{x - x_0}{x_1 - x_0}M(P_{10}) + \frac{x_1 - x}{x_1 - x_0}M(P_{00})\right)$$



Localisation : Optimisation stochastique

- Recherche d'un score maximum par échantillonnage aléatoire
- Stratégie la plus basique :
  - Initialisation avec la position de l'odométrie
  - Répéter N fois (ou répéter tant que moins de N tirages sans amélioration)
    - Tirer une position autour de la meilleure estimation avec un bruit gaussien $(0,\sigma)$
    - Estimer le score pour cette position
    - Si le score est meilleur, conserver cette position
  - Attention au choix de σ, du critère d'arrêt

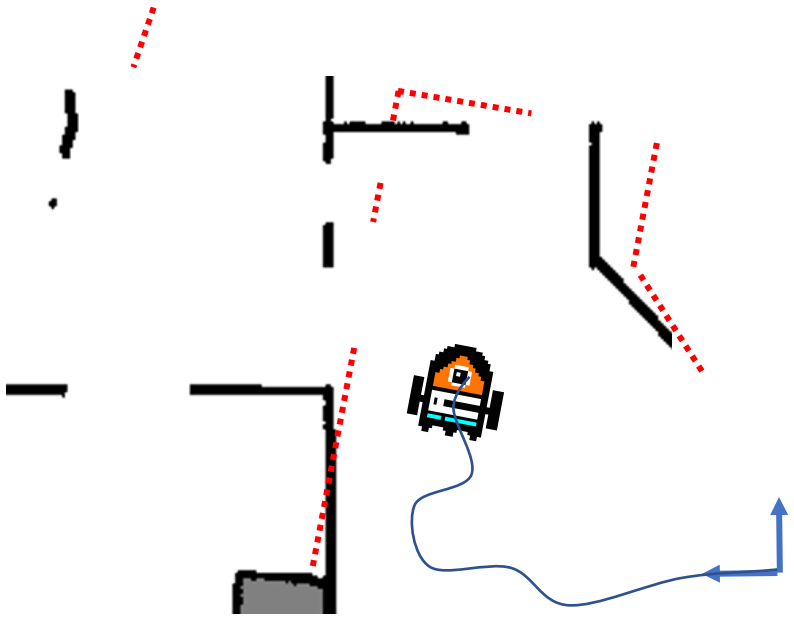


σ faible, risque de minimum local

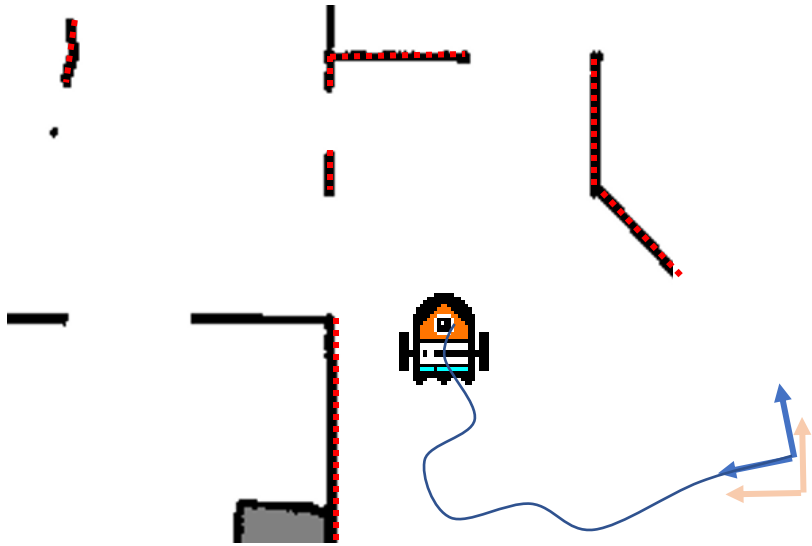
σ fort, convergence lente

Localisation : Optimisation stochastique

- En pratique, on met à jour l'origine du repère odométrie dans la carte
  - Permet une localisation à la fréquence de l'odométrie (~ 100Hz), avec des corrections à plus basse fréquence (~10Hz)



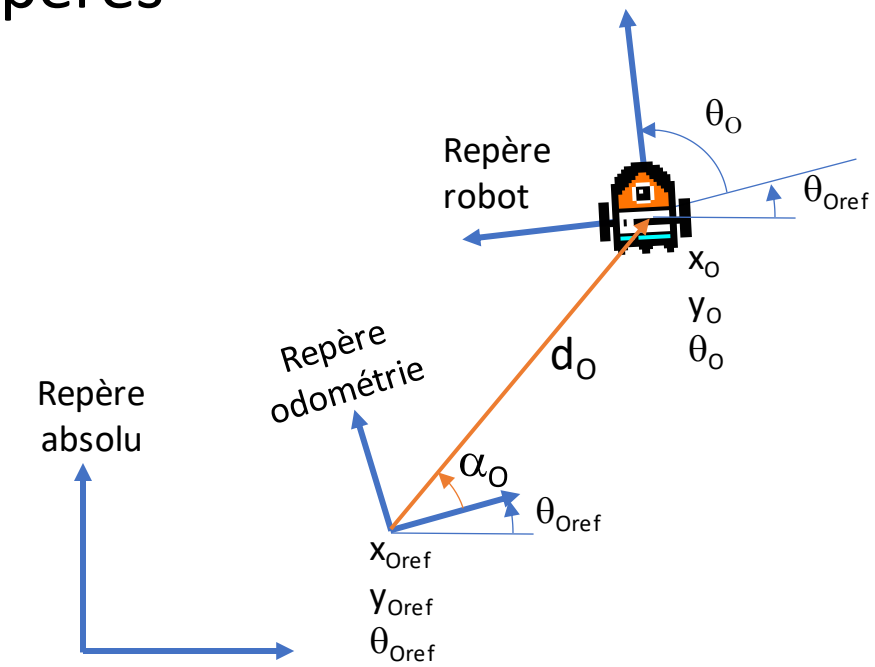
Repère odométrie courant



Repère odométrie corrigé

Localisation : Optimisation stochastique

- Repères



(x, y, θ) : Position du robot dans le repère absolu

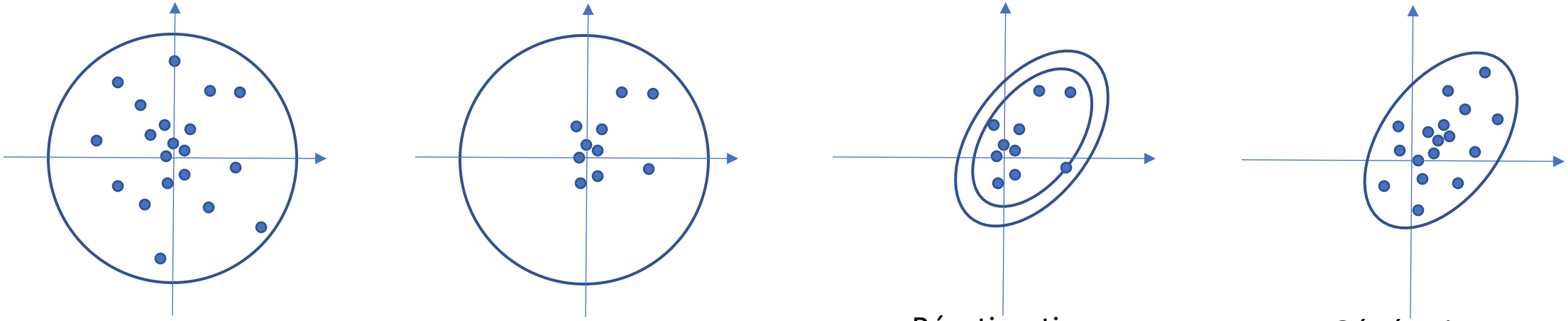
(x_O, y_O, θ_O) : Position du robot dans le repère odométrie

$(x_{Oref}, y_{Oref}, \theta_{Oref})$: Position du repère odométrie dans le repère absolu

$x = x_{Oref} + d_O * \cos(\theta_{Oref} + \alpha_O)$
$y = y_{Oref} + d_O * \sin(\theta_{Oref} + \alpha_O)$
$\theta = \theta_{Oref} + \theta_O$

Méthodes plus avancées - CEM

- Cross Entropy Method (CEM)
  - Génère une population de solutions suivant une loi Gaussienne
  - Réestime les paramètres de Gaussienne pour population suivante
  - Répéter
    - Générer N points selon une Gaussienne (μ_t, σ_t) et évaluer ces points
    - Estimer $(\mu_{t+1}, \sigma_{t+1})$ à partir des N_e meilleurs points, ajouter ε à σ_{t+1}



N solutions générées

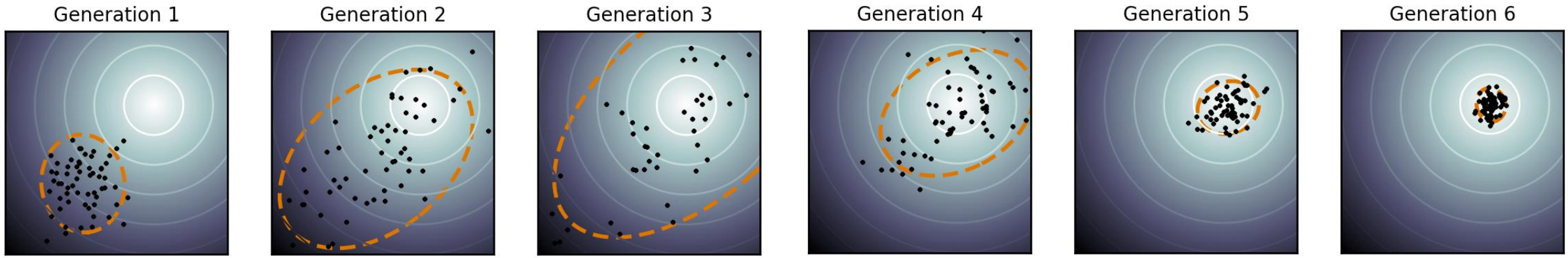
N_e meilleures solutions

Réestimation Moyenne/variance

Génération suivante

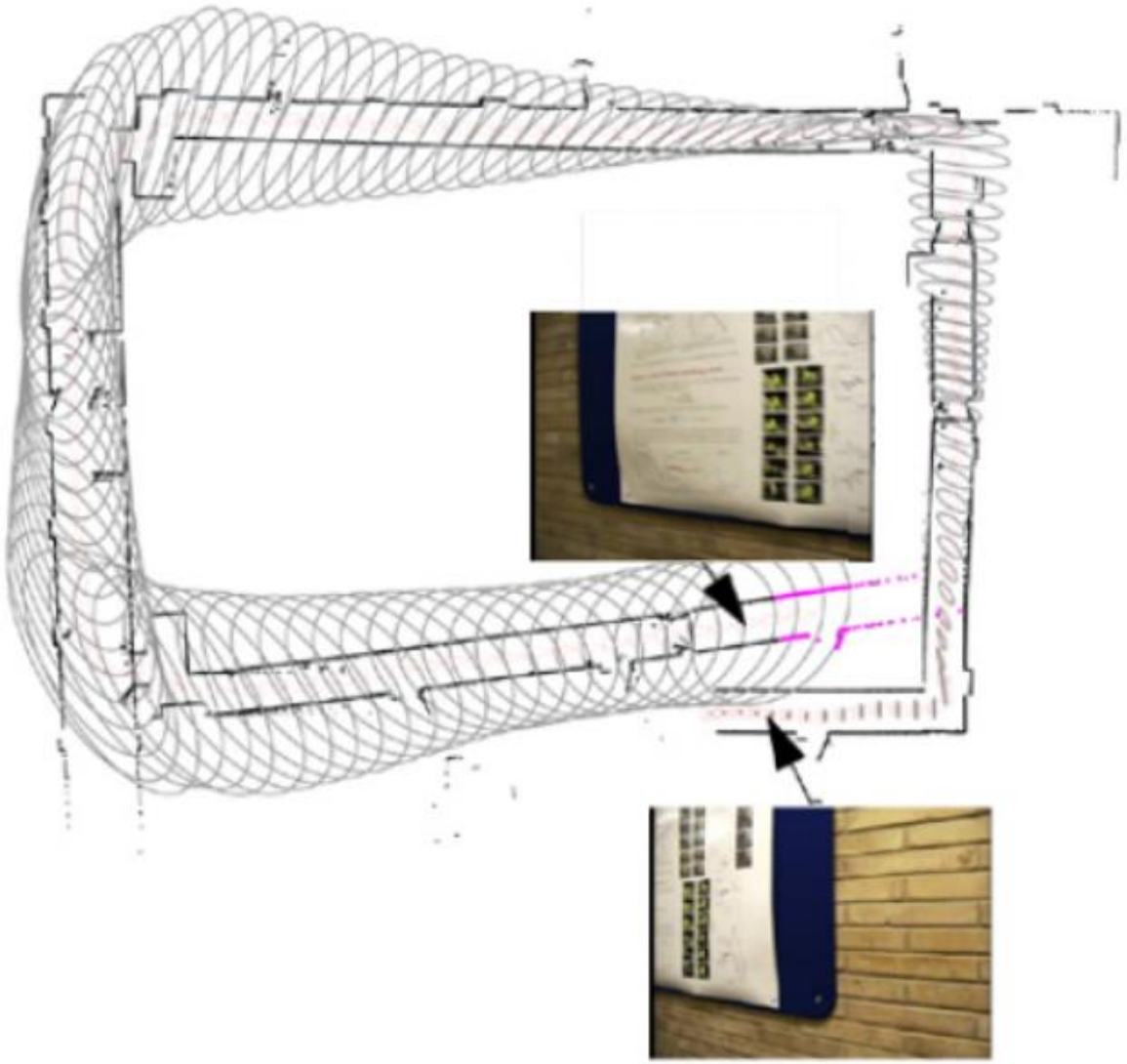
Méthodes plus avancées – CMA-ES

- Covariance Matrix Adaptation Evolution Strategy (CMA-ES)
  - Algorithme plus complexe de mise à jour de la Gaussienne
  - Prise en compte l'historique des mises à jour
  - Cf https://en.wikipedia.org/wiki/CMA-ES



SLAM

- Cartographie et Localisation Simultanées (SLAM)
  - Carte mise a jour à partir du résultat de localisation
  - Erreur de localisation s'intègre dans la carte
  - Difficulté pour les grands environnements cycliques
  - Cartographie erronée

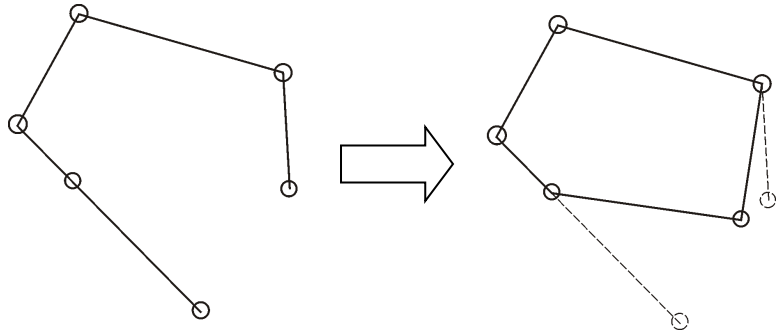


P. Newman, Oxford Mobile Robotics Group

SLAM

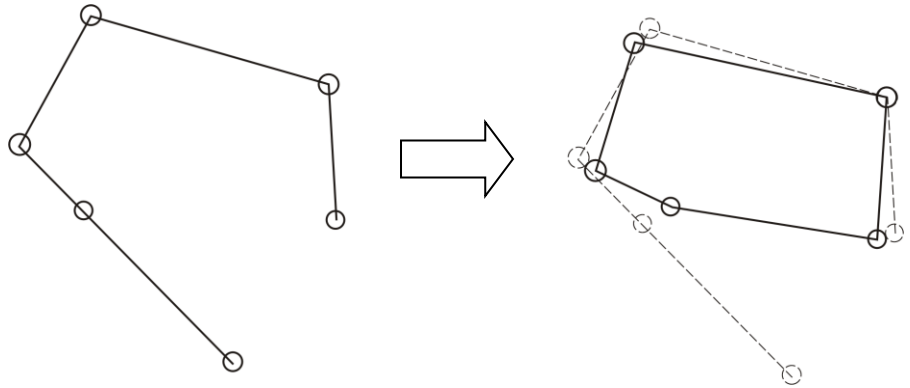
Cartographie incrémentale

- Pas de reprise des informations passées si elle se révèlent erronées
- Modifications locales de la carte
- Cas de ROB201



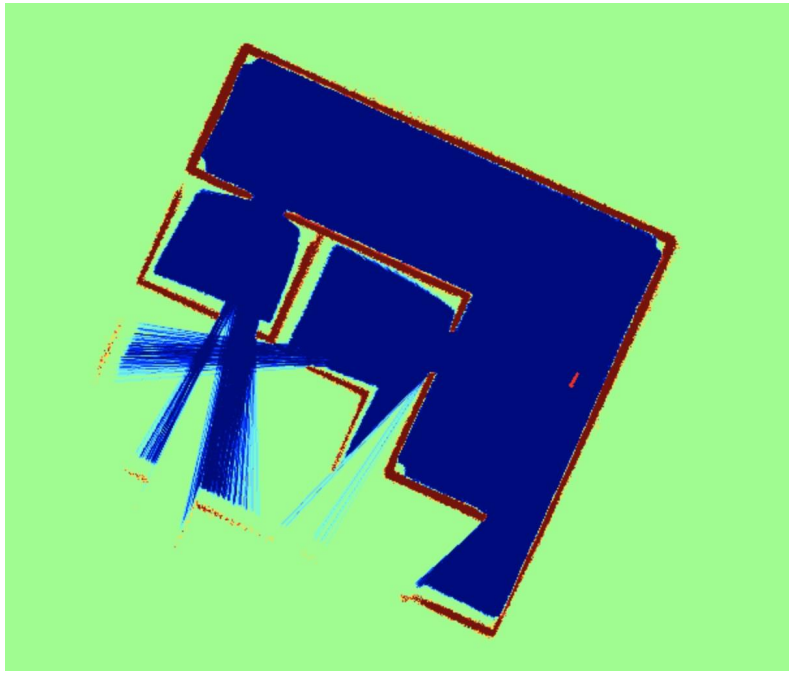
Cartographie avec retour en arrière

- Lorsque la localisation se révèle fausse, propagation des erreurs dans la carte
- Possibilité de modifications globales
- Cf ROB312



SLAM

- Approche simple du SLAM:
  - Initialiser la carte avec le premier scan
  - Répéter:
    - Localiser le robot par rapport à cette carte
    - Si la localisation est précise, mettre à jour la carte



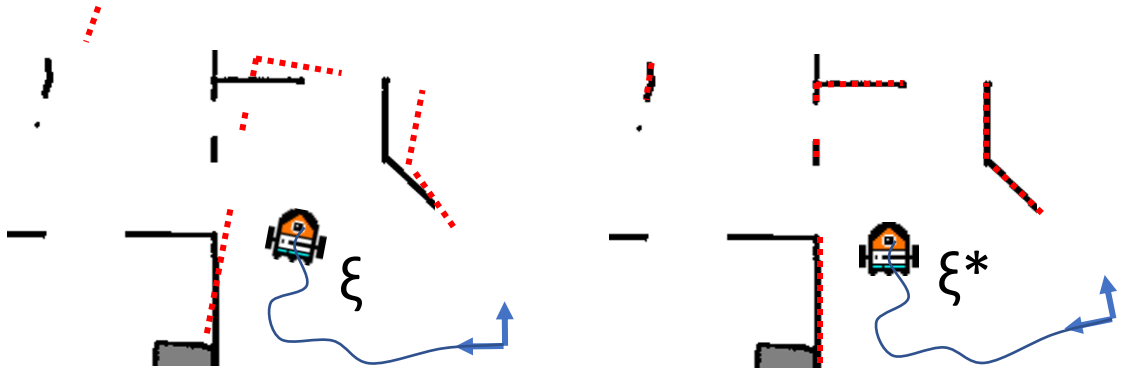
- Méthode proche de 'TinySLAM'

  Steux, B., & El Hamzaoui, O. (2010, December). tinySLAM: A SLAM algorithm in less than 200 lines C-language program. In *2010 11th International Conference on Control Automation Robotics & Vision*

- Et de 'HectorSLAM'

  S. Kohlbrecher and J. Meyer and O. von Stryk and U.Klingauf : A Flexible and Scalable SLAM System with Full 3D Motion Estimation.

HectorSLAM



Localisation par rapport à la carte par descente de gradient

- Trouver la pose ξ* tq tous les pts du laser soient dans des cases occupées:

Minimisation sur la somme des points pour prendre en compte bruits et erreurs

Position absolue du point laser i pour une localisation ξ donnée

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \overbrace{\sum_{i=1}^{n}} [1 - M(\mathbf{S}_i(\boldsymbol{\xi}))]^2$$

Valeur de la carte aux coordonnées données par S_i (≤ 1)

- Dvp de Taylor ordre 1 de la fonction M : $\sum_{i=1}^{n} [1 - M(\mathbf{S}_i(\boldsymbol{\xi} + \Delta\boldsymbol{\xi}))]^2 \rightarrow 0$

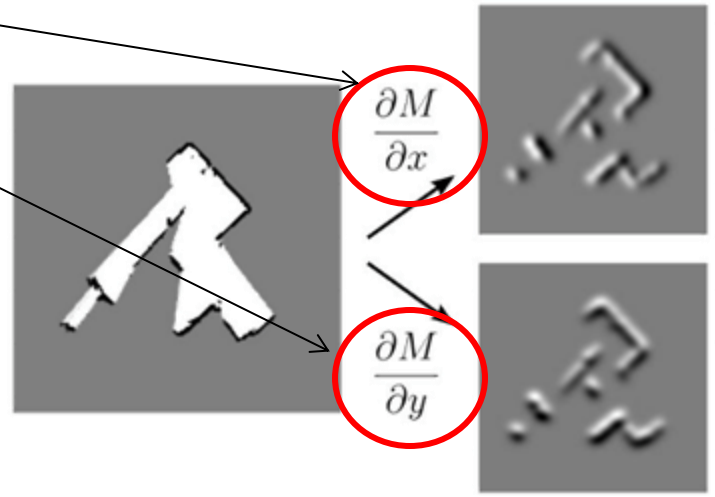
HectorSLAM

- Méthode de Gauss-Newton (minimisation d'une somme de fonctions carrées) : donne Δξ minimisant la fonction objectif

H matrice Hessienne de la carte ou approximation

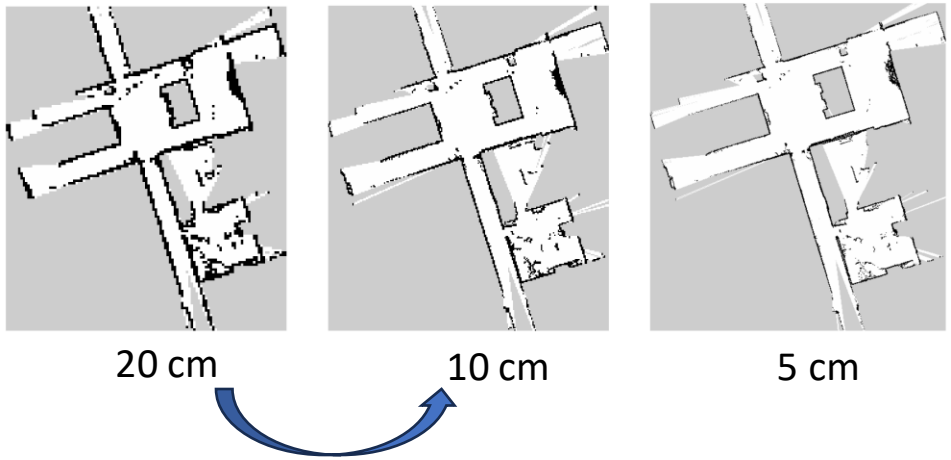
$$\Delta\boldsymbol{\xi} = \widetilde{\mathbf{H}^{-1}} \sum_{i=1}^{n} \left[\nabla M(\mathbf{S}_i(\boldsymbol{\xi})) \frac{\partial \mathbf{S}_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \right]^T [1 - M(\mathbf{S}_i(\boldsymbol{\xi}))]$$

Gradient de la carte



HectorSLAM

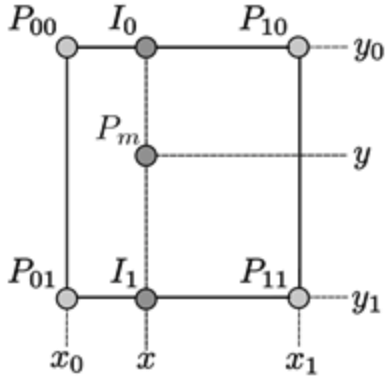
- Quelques trucs supplémentaires:
  - Approche multi-échelle



Estimation de Pose dans la carte de résolution
20 cm donnée en entrée de celle de 10 cm

  - Estimation continue via interpolation.

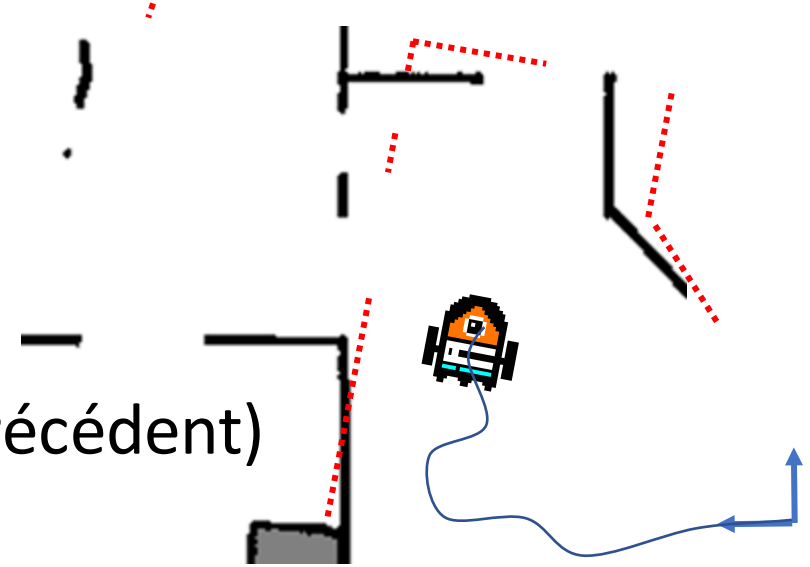
$$M(P_m) \approx \frac{y - y_0}{y_1 - y_0}\left(\frac{x - x_0}{x_1 - x_0}M(P_{11}) + \frac{x_1 - x}{x_1 - x_0}M(P_{01})\right) + \frac{y_1 - y}{y_1 - y_0}\left(\frac{x - x_0}{x_1 - x_0}M(P_{10}) + \frac{x_1 - x}{x_1 - x_0}M(P_{00})\right)$$



HectorSLAM

https://www.youtube.com/watch?v=sieBqVxTz2c

Notre algorithme de SLAM



- Initialiser la carte avec le premier scan (TP précédent)
- A chaque mesure
  - Calculer la position avec la référence + odométrie
  - Estimer le score avec cette position
  - Répéter tant que moins de N tirages sans amélioration
    - Tirer une nouvelle référence odométrie autour de la référence précédente avec un bruit gaussien $(0,\sigma)$
    - Calculer la position avec la nouvelle référence + odométrie
    - Estimer le score pour cette position
    - Si le score est meilleur, conserver cette référence
  - Si score > seuil, mettre à jour la carte avec la mesure laser (TP précédent)

TP 04

- Ecrire la fonction `get_corrected_pose` de la classe `TinySlam`
- Ecrire la fonction `score` de la classe `TinySlam`
- Ecrire la fonction `localise` de la classe `TinySlam` avec une recherche aléatoire simple
- Intégrer avec la cartographie pour boucler le SLAM
- Extensions possibles : implémenter CEM, CMA-ES, interpolation bilinéaire