UNIVERSITÉ de Picardie Jules Verne

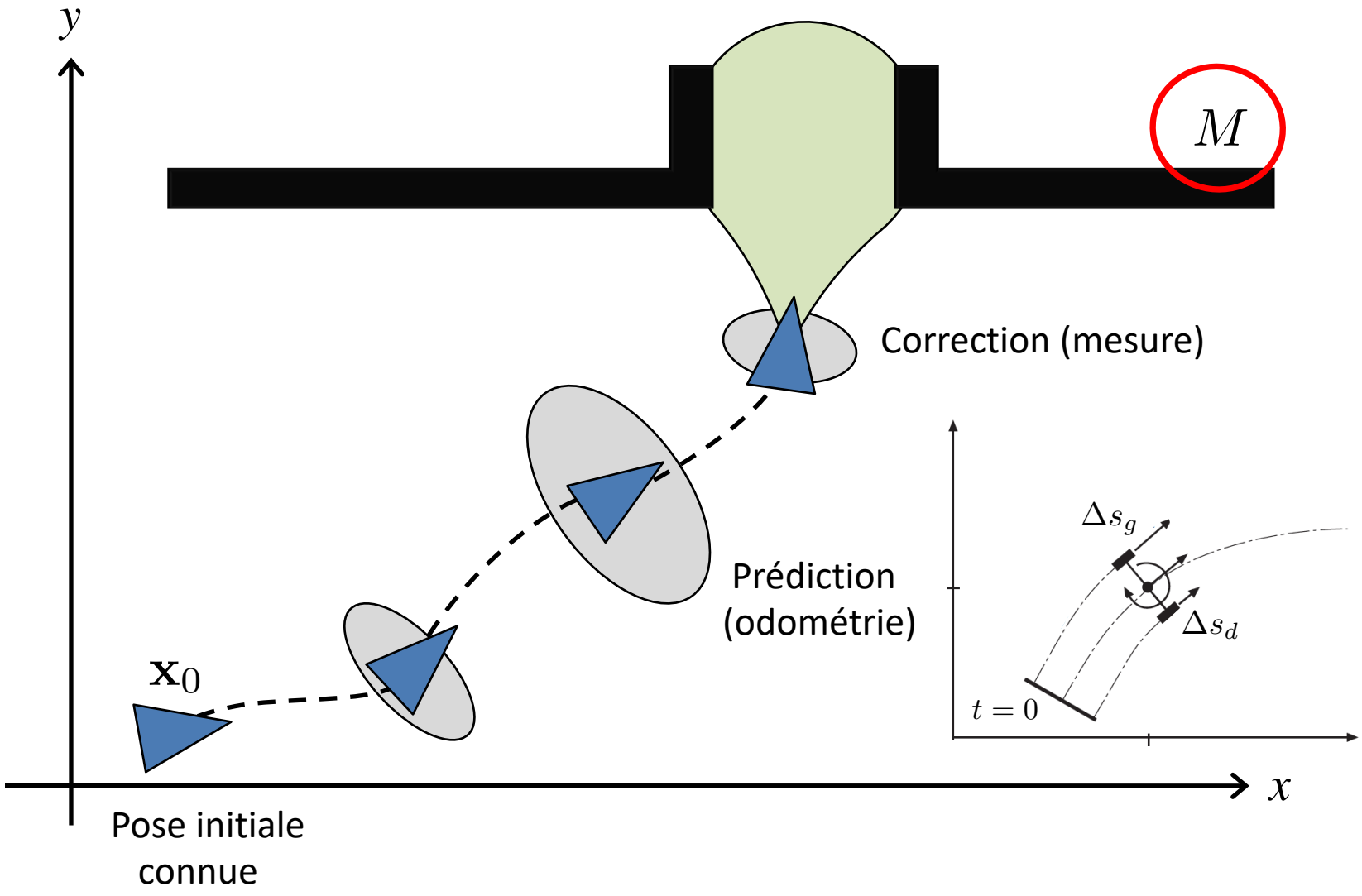
Localisation et navigation de robots

TD2: Localisation par filtre de Kalman étendu

Fabio MORBIDI

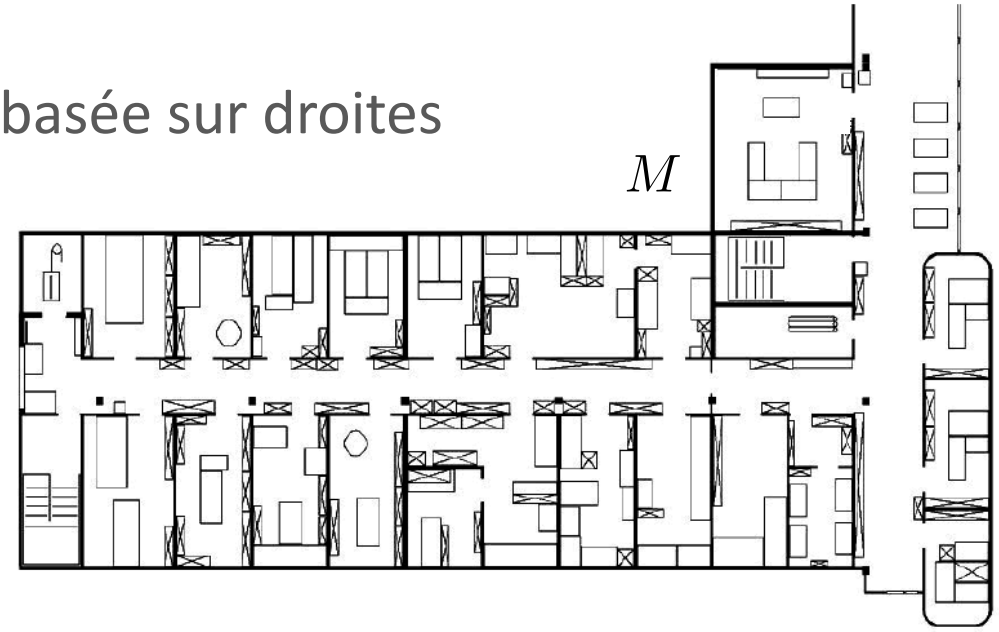
2022-2023

Localisation d'un robot mobile basée sur carte



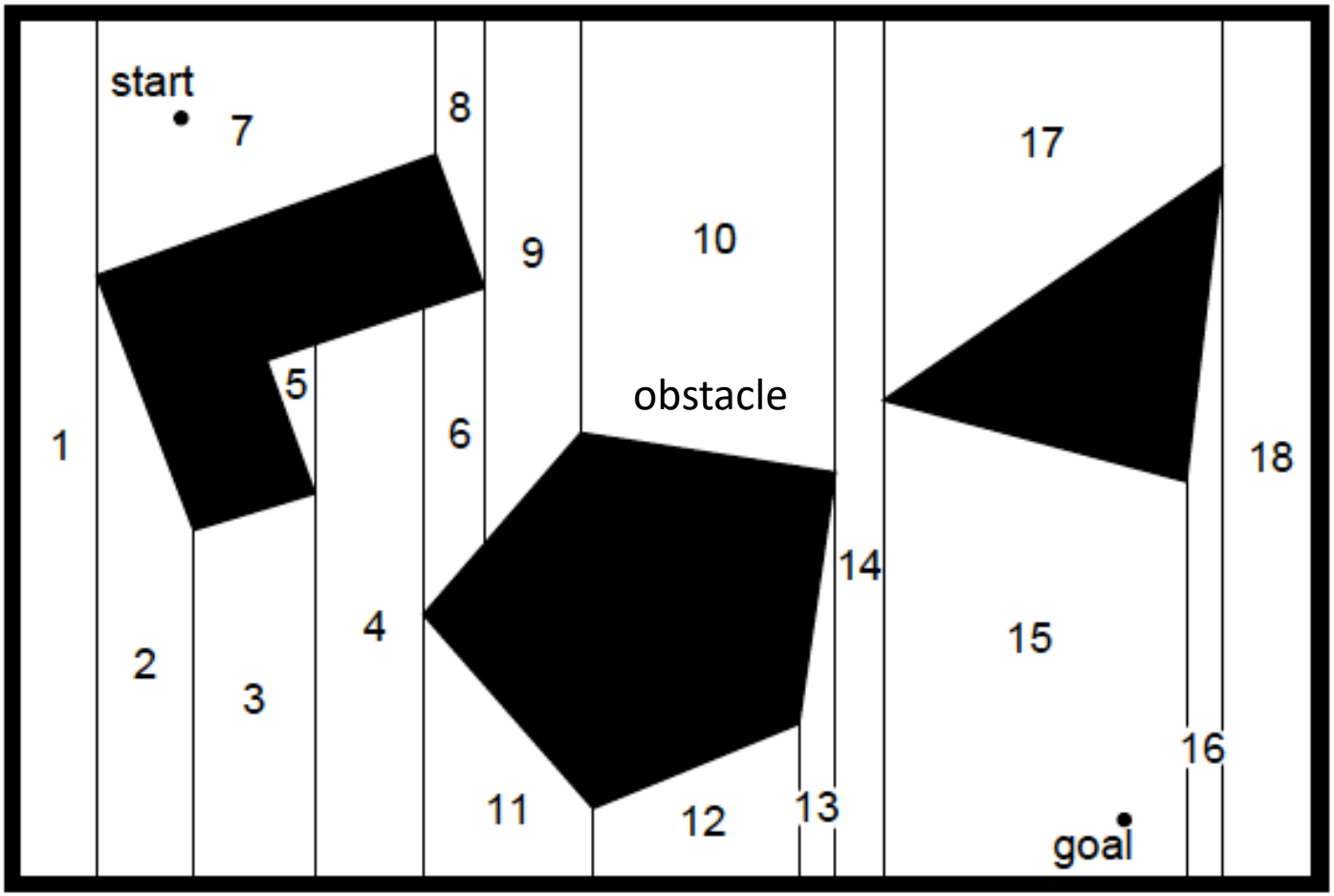
Représentation de la carte

1. Décomposition *exacte* en cellules
2. Décomposition *approchée* en cellules
   - Cellules de taille fixe
   - Cellules de taille variable
3. Cartes topologiques
4. Représentation continue basée sur droites



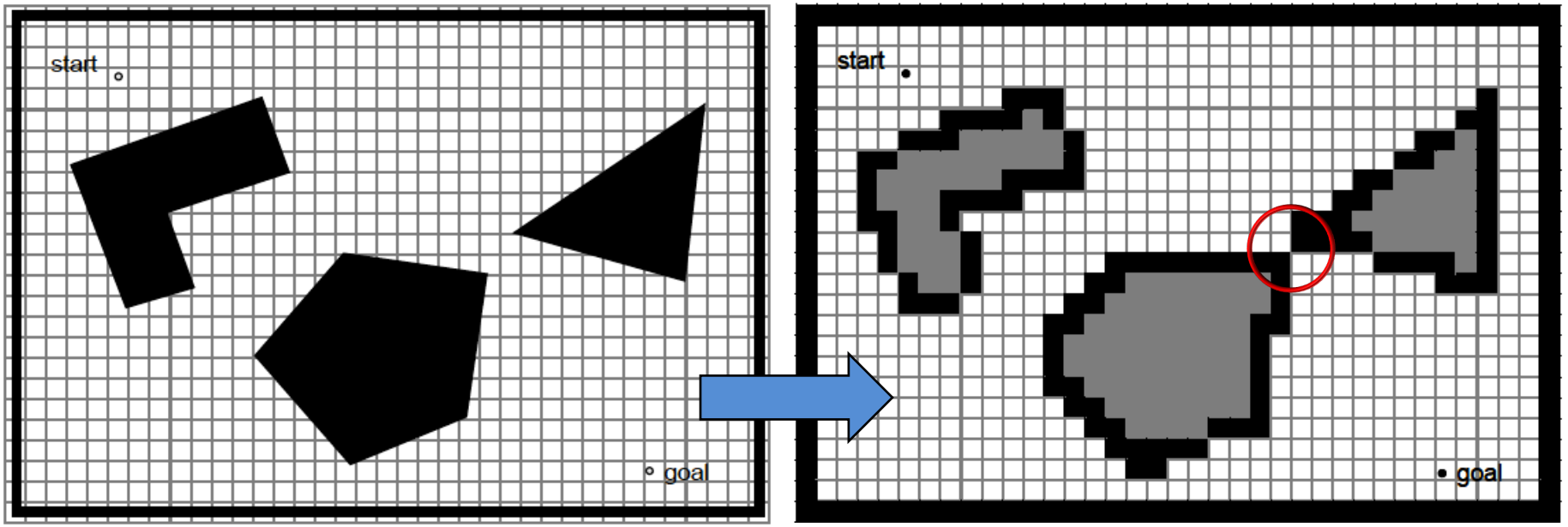
Représentation de la carte

1. Décomposition exacte en cellules [Latombe, 1991]



Représentation de la carte

2. Décomposition approchée en cellules
   - Cellules de taille fixe
     - Simple et très utilisée (« occupancy grid », en anglais)
     - Les passages étroits disparaissent (la représentation n'est pas complète)
     - Grande mémoire nécessaire si la taille des cellules est petite

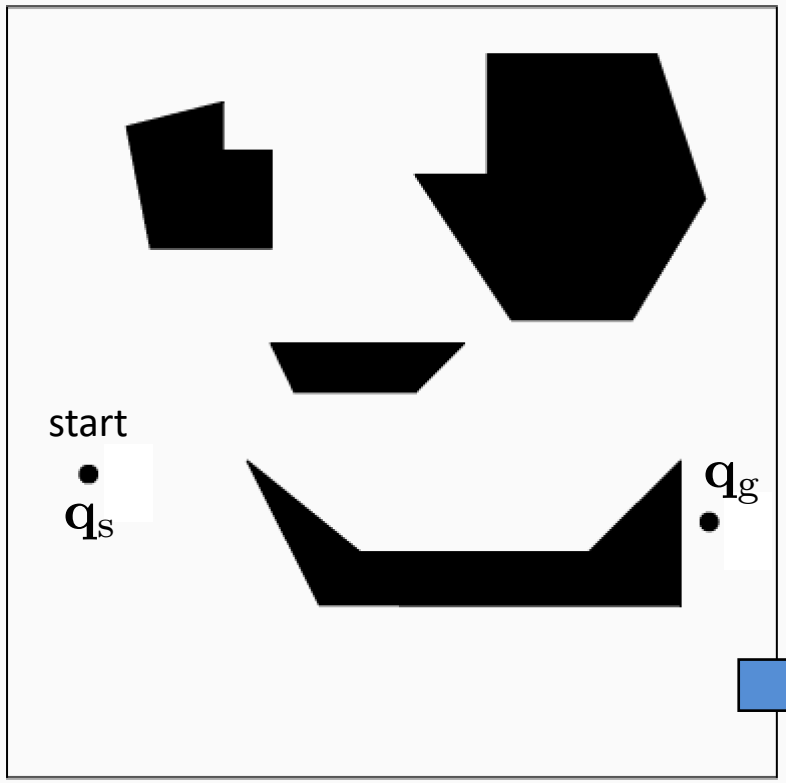


Environnement continu

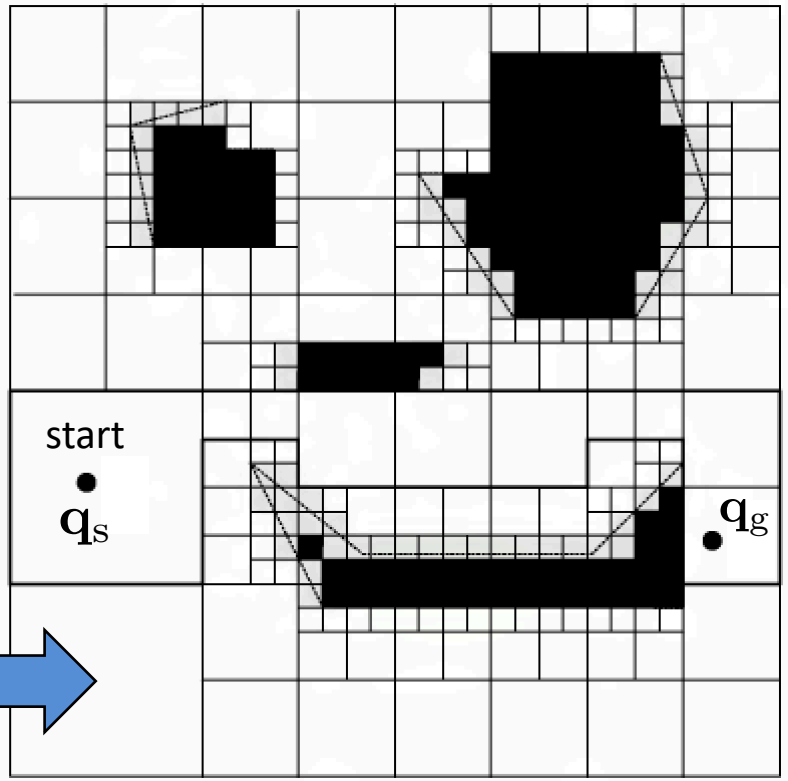
Approximation discrète de l'environnement

Représentation de la carte

2. Décomposition approchée en cellules
   - Cellules de taille variable (décomposition adaptative)
     - Réduction du nombre de cellules et de l'espace de mémoire nécessaire



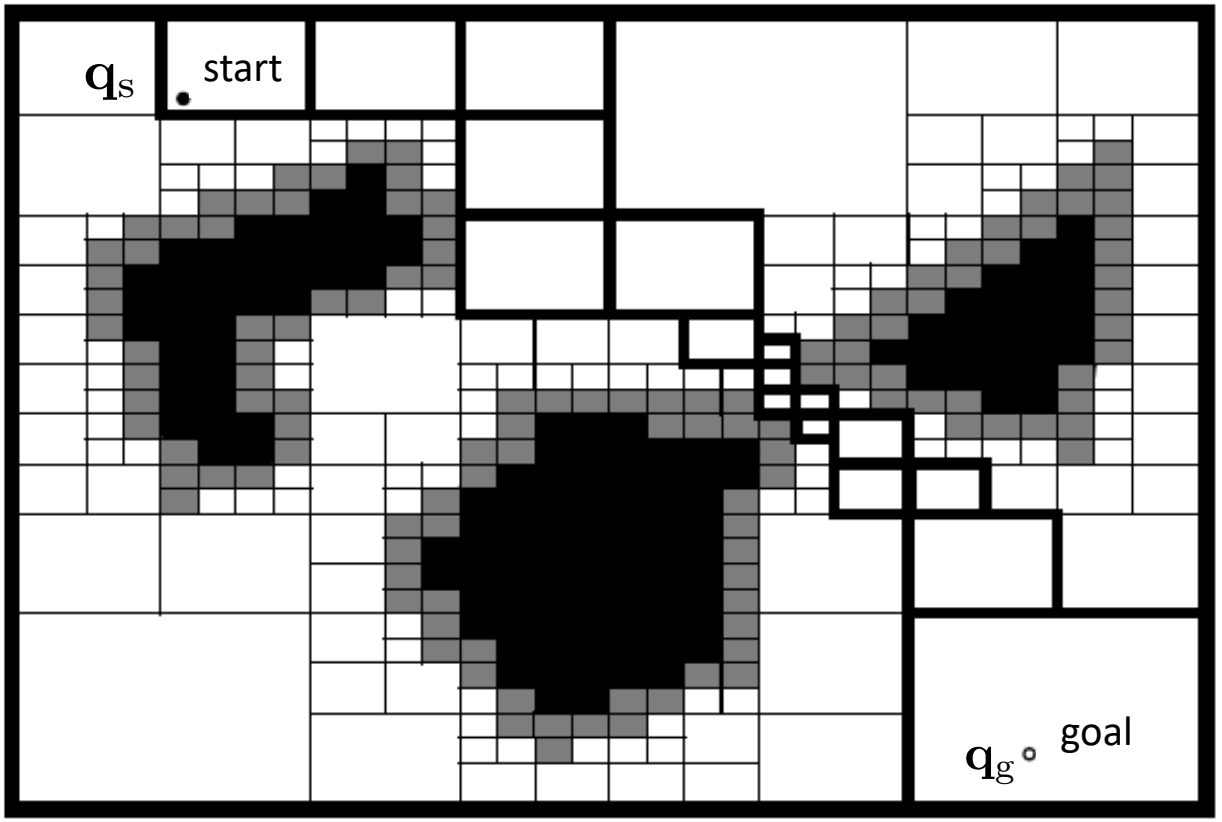
Environnement



Décomposition adaptative de l'environnement

Représentation de la carte

2. Décomposition approchée en cellules

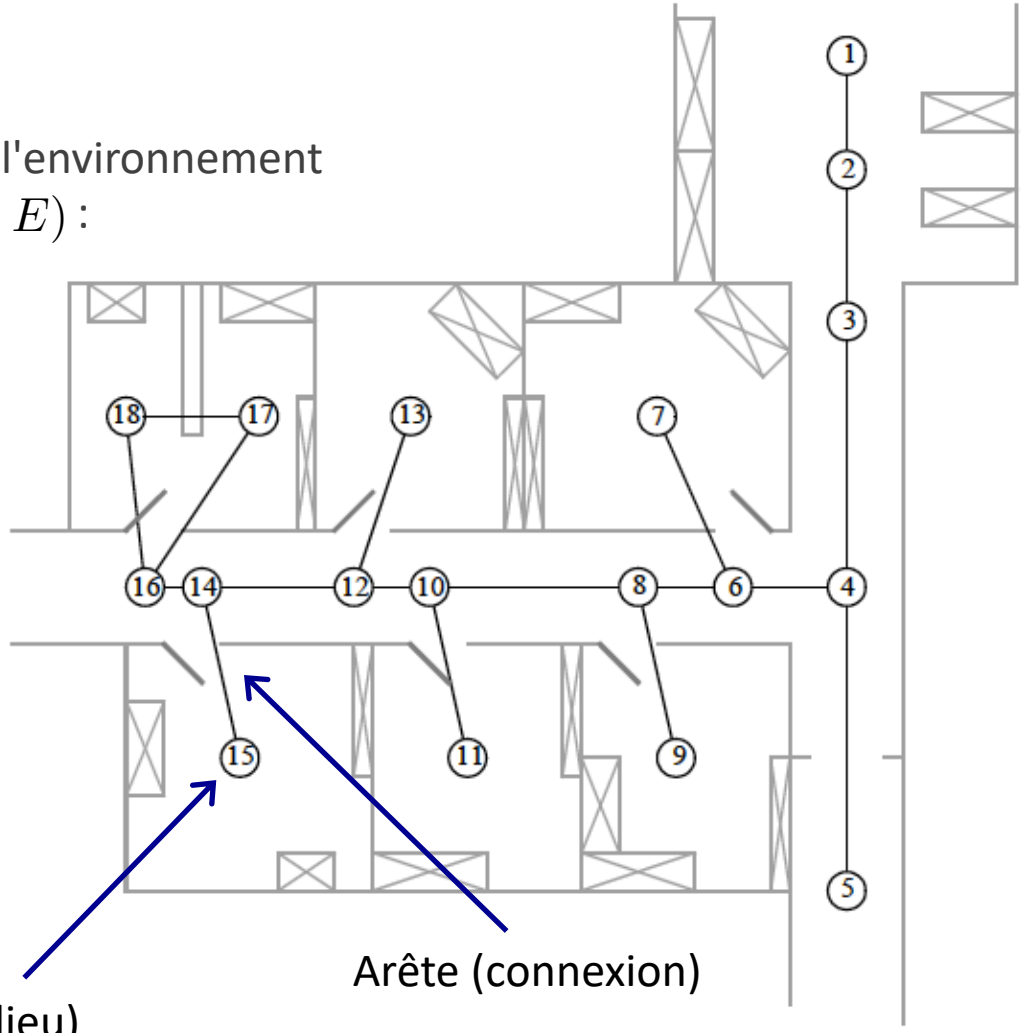


Remarque:
En informatique, un **quadtree** ou **arbre quaternaire** est une structure de données de type arbre dans laquelle chaque nœud a quatre fils

Représentation de la carte

3. Cartes topologiques

- Les cartes topologiques représentent l'environnement comme un graphe non orienté $\mathcal{G} = (V, E)$:
  - Les **nœuds** V correspondent à l'*espace géométrique*
  - Les **arêtes** E correspondent aux *connexions* physiques entre les nœuds
- Les cartes topologiques sont dépourvues *d'**échelle*** et de ***distance***, mais les *relations topologiques* (par ex. gauche, droite, nord, sud) sont conservées



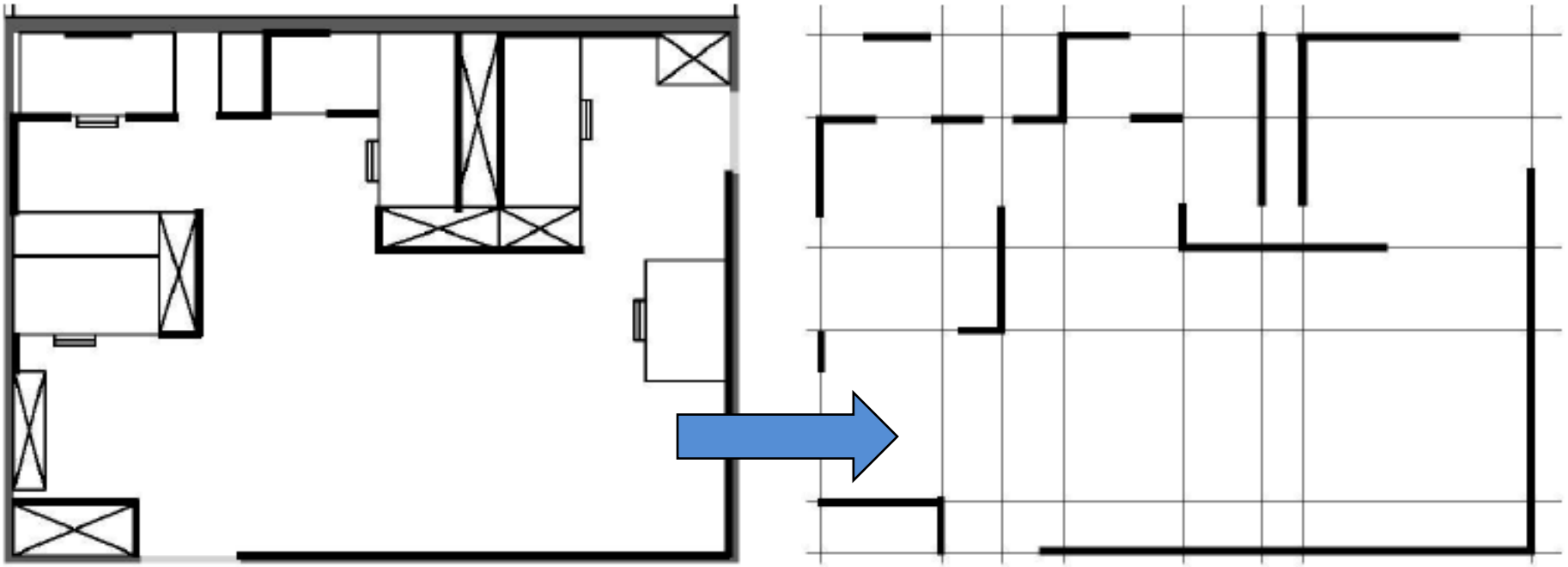
Représentation de la carte

- Exemple de carte topologique
  - Carte du *métro de Paris*

Représentation de la carte

4. Représentation continue basée sur droites

a) Représentation avec un ensemble de droites (finies ou infinies)
b) Extraction de droites à partir de mesures laser, sonar ou d'images
   - Transformée de Hough
   - RANSAC (*RAndom Sample Consensus*) [Fischler & Bolles, 1987]
   - *Split-and-merge* [Pavlidis & Horowitz, 1974]



Environnement

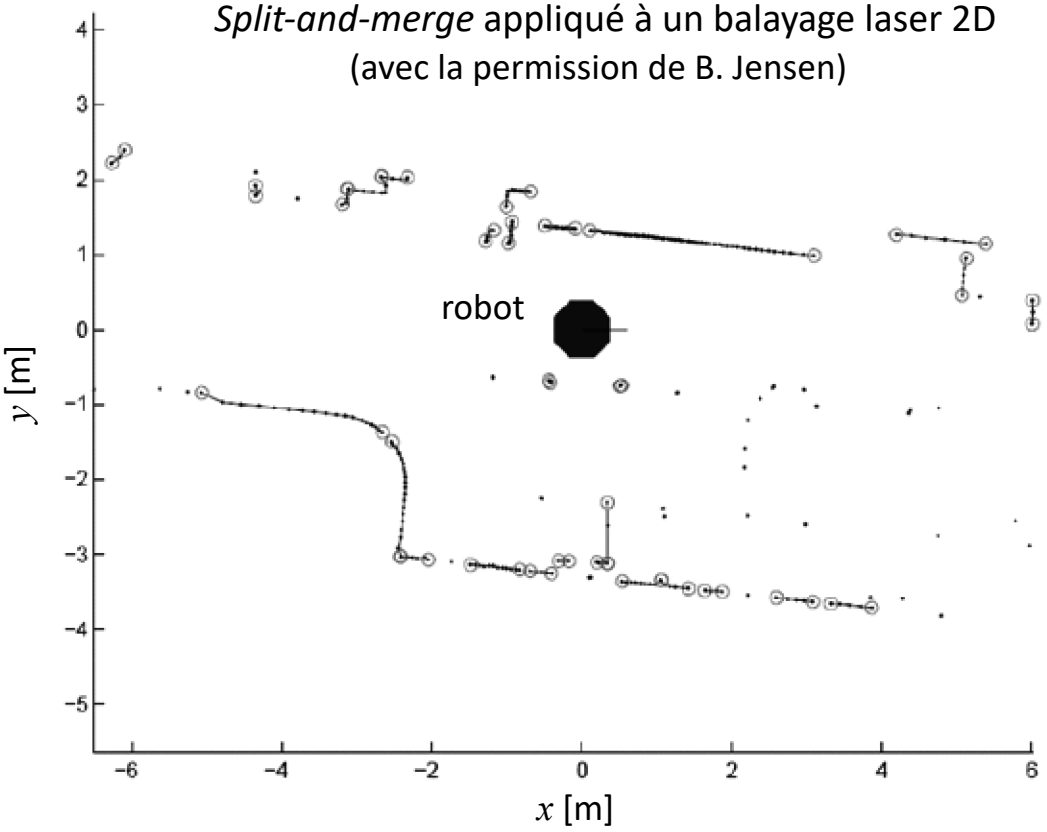
Représentation avec droites infinies

Représentation de la carte

4. Représentation continue basée sur droites

- Algorithme incrémental [Forsyth & Ponce, 2003]
- Régression linéaire [Arras & Siegwart, 1997]
- Algor. Espérance-Maximisation [Forsyth & Ponce, 2003] (inconvénient: minima locaux)

Split-and-merge appliqué à un balayage laser 2D
(avec la permission de B. Jensen)



Représentation de la carte

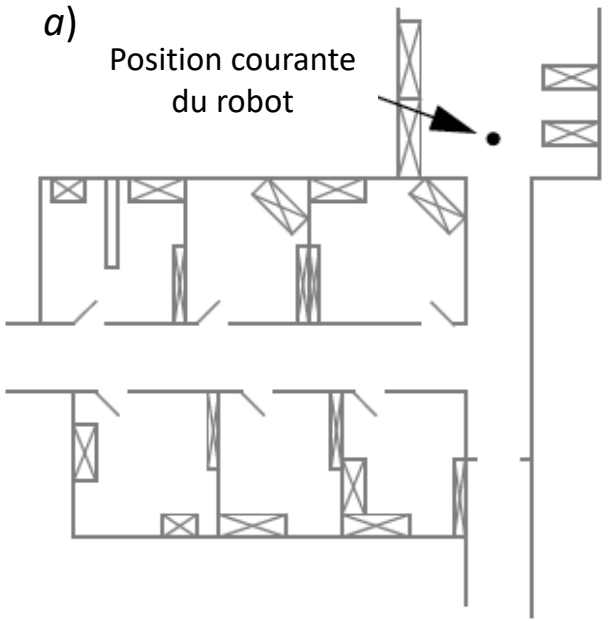
4. Représentation continue basée sur droites

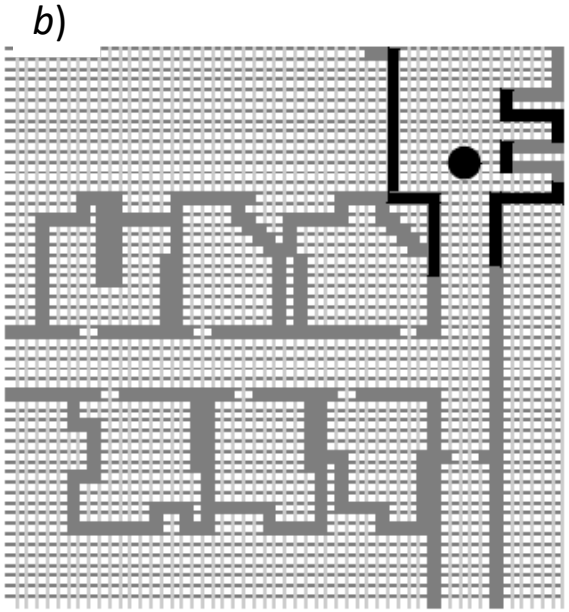
	Déterministe	Fréquence [Hz]	Faux positifs [%]	Précision
Transfor. de Hough	Oui	10	30	++++
RANSAC	Non	30	30	++++
Split-and-merge	Oui	1500	10	+++
Alg. incrémental	Oui	600	6	+++
Régression linéaire	Oui	400	10	+++
Alg. Espér. - Maxim.	Non	1	50	++++

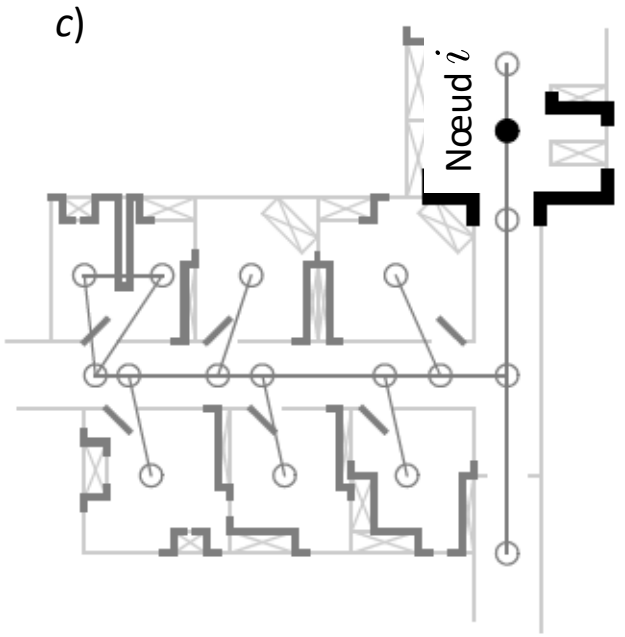
Comparaison des six algorithmes pour l'extraction de droites*
à partir de données laser 2D

* "*A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics*", V. Nguyen, A. Martinelli, N. Tomatis, R. Siegwart, In Proc. IEEE/RSJ Int. Conf. Intel. Robots and Systems, pp. 1929-1934, 2005

Représentation de la carte: sommaire





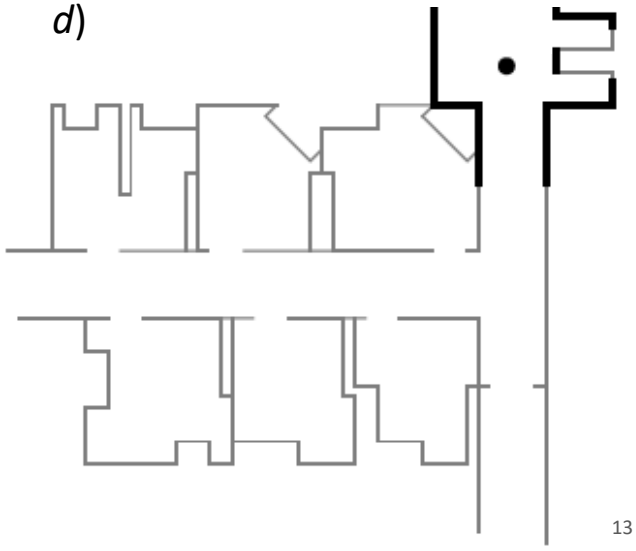


a) Carte “réelle” avec murs, mobilier, portes, etc.

b) *Décomposition approchée*: environ 3000 cellules de taille fixe 50 cm × 50 cm. Chaque cellule est pleine (présence d'un obstacle) ou vide (espace libre)

c) *Carte topologique* avec 18 nœuds et 17 arêtes

d) *Représentation continue basée sur droites*: 100 droites avec 2 paramètres



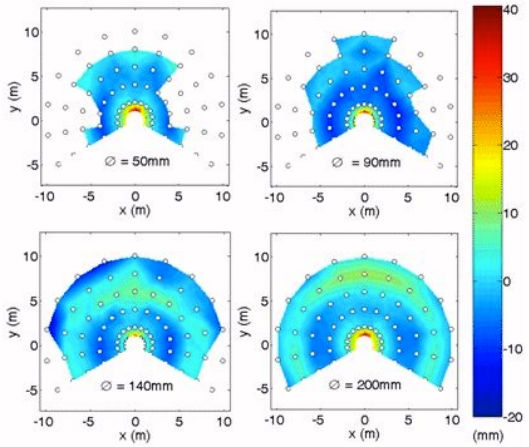
Problème de localisation probabiliste

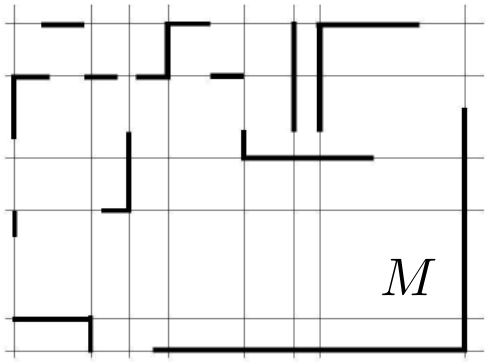
Approche probabiliste au problème de localisation (incrémentielle)

Calcul de la croyance sur l'état du robot (*fonction de densité de probabilité ou pdf*) à chaque instant de temps

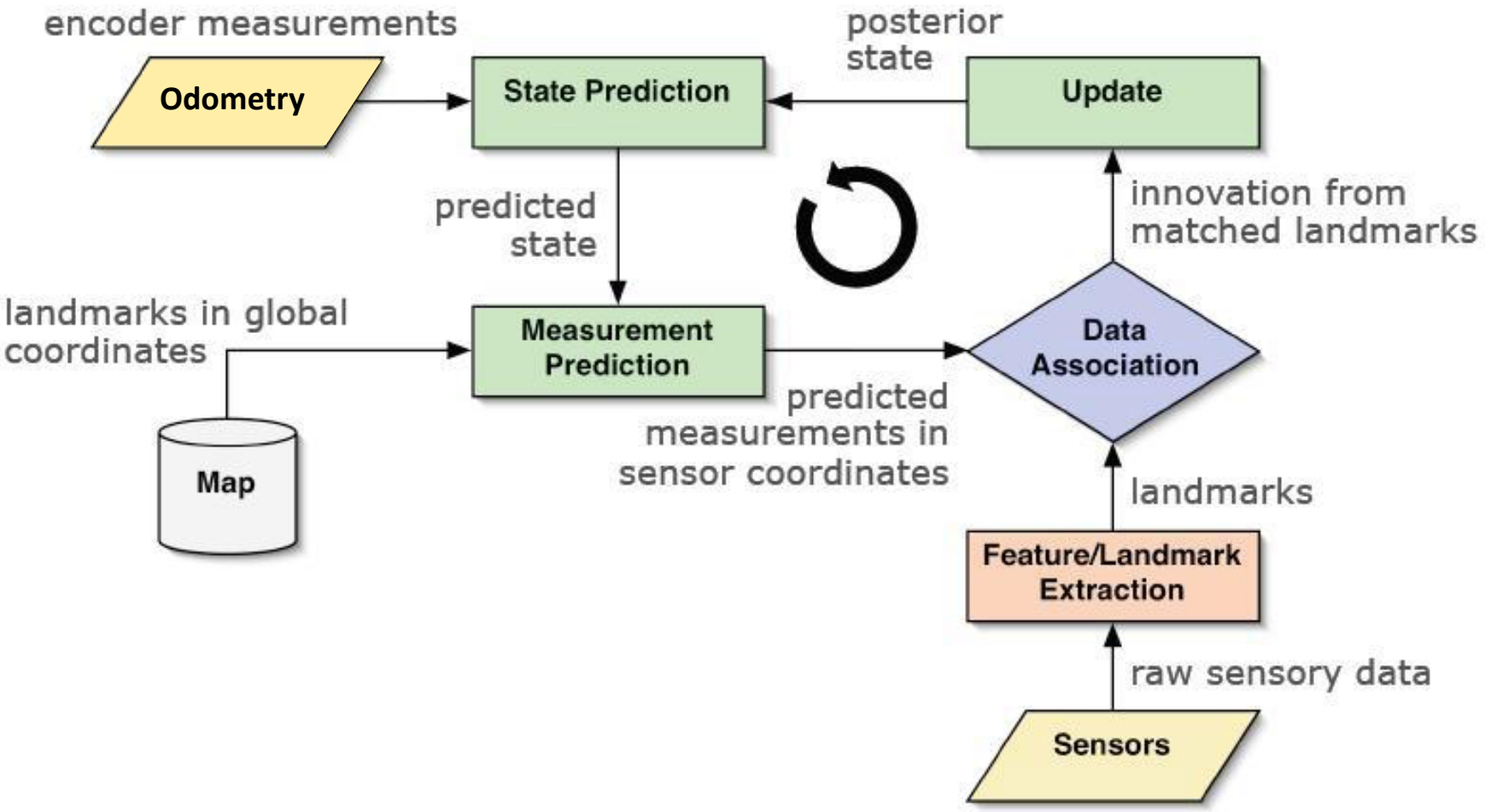
Ingrédients nécessaires:

1. Densité de probabilité *initiale*: $\mathrm{croyance}(\mathbf{x}_0)$
2. Modèle d'erreur statistique des *capteurs proprioceptifs* (par ex. encodeurs des roues du robot)
3. Modèle d'erreur statistique des *capteurs extéroceptifs* (par ex. lasers, sonars, caméras)
4. Carte de l'environnement M





Fusion des mesures proprioceptives et extéroceptives

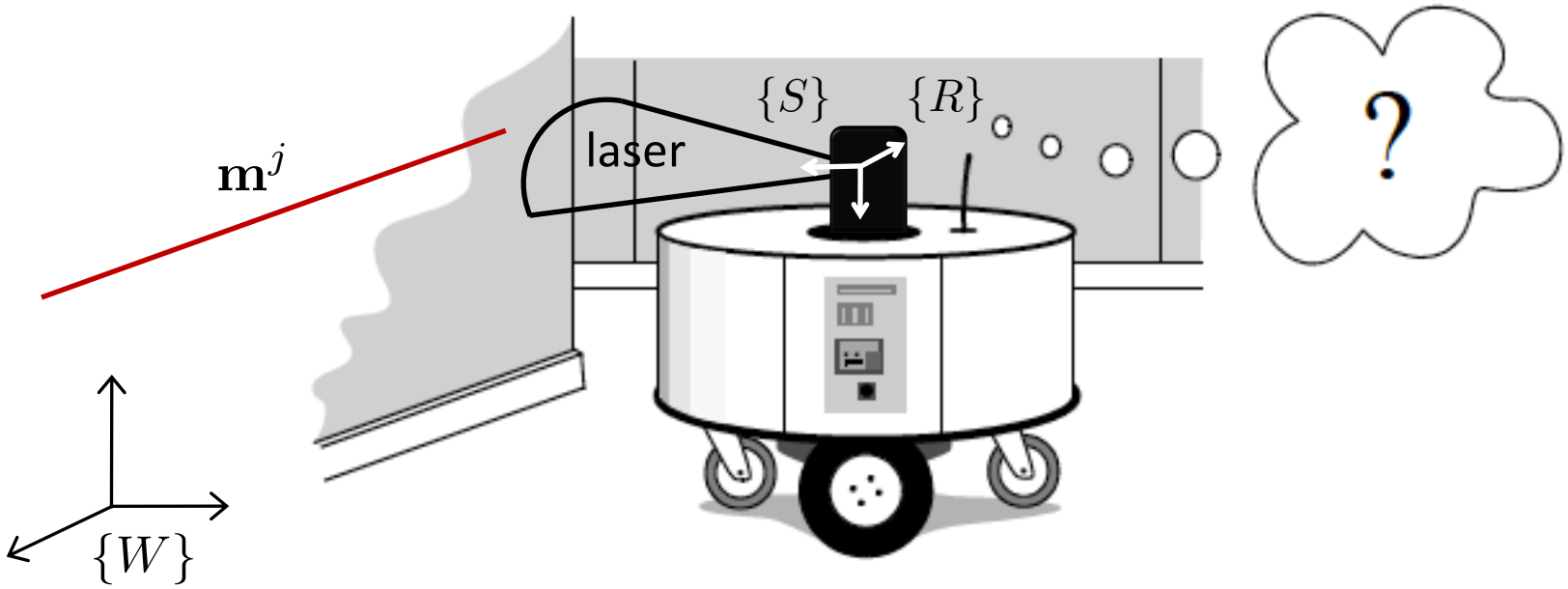


Landmark = balise (ou amer) passive

Localisation d'un robot par EKF

Nos assumptions:

- Robot de type **unicycle** à **conduite différentielle**
- Capteurs: **encodeurs** sur les roues (proprioceptifs), **télémètre laser** (extéroceptif)
- Le repère du télémètre laser $\{S\}$ coïncide avec le repère du robot $\{R\}$
- Primitives $\mathbf{m}^j$ de M : **droites** (exprimées dans le repère monde $\{W\}$)



Localisation d’un robot par EKF

① Prédiction

② Correction

a) Observation des primitives par mesure capteurs

b) Prédiction de mesure des primitives

c) Appariement des mesures

d) Estimation

Localisation d’un robot par EKF

① Prédiction

② Correction

a) Observation des primitives par mesure capteurs

b) Prédiction de mesure des primitives

c) Appariement des mesures

d) Estimation

① Prédiction

Prédiction de la nouvelle pose à partir de:

- L'ancienne pose: $\mathbf{x}_{t-1} = [x_{t-1}, y_{t-1}, \theta_{t-1}]^T$
- Le déplacement des roues (gauche, droite): $\mathbf{u}_t = [\Delta s_g, \Delta s_d]^T$

On peut utiliser le modèle pour l'exactitude odométrique développé dans le TD1:

$$\widehat{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_d + \Delta s_g}{2} \cos\left(\theta_{t-1} + \frac{\Delta s_d - \Delta s_g}{2L}\right) \\ \frac{\Delta s_d + \Delta s_g}{2} \sin\left(\theta_{t-1} + \frac{\Delta s_d - \Delta s_g}{2L}\right) \\ \frac{\Delta s_d - \Delta s_g}{L} \end{bmatrix}$$

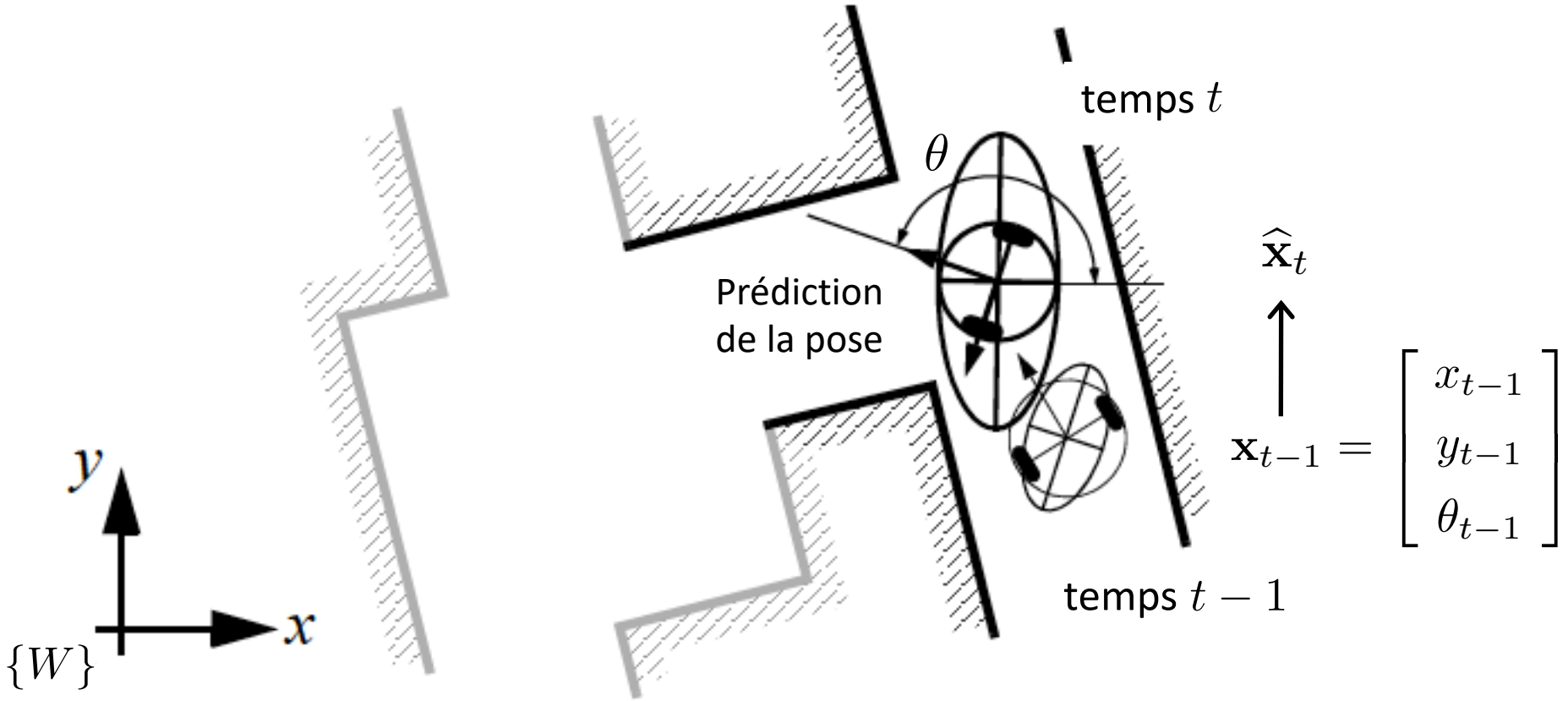
Modèle non linéaire !

$$\widehat{\mathbf{P}}_t = \mathbf{F}_\mathbf{x} \mathbf{P}_{t-1} \mathbf{F}_\mathbf{x}^T + \mathbf{F}_\mathbf{u} \mathbf{Q}_t \mathbf{F}_\mathbf{u}^T$$

où $\mathbf{P}_{t-1}$ covariance de l'état précédent du robot $\mathbf{x}_{t-1}$

$\mathbf{Q}_t$ covariance du bruit du modèle de mouvement: $\mathbf{Q}_t = \text{diag}(k_d|\Delta s_d|, k_g|\Delta s_g|)$

① Prédiction



Rappel le TD1 - Exercices 1 et 2

Fonctions «MAJEtatOdometrie» et «*propageErreurs*» pour la mise à jour de la pose du robot et le calcul de la matrice de covariance $\boldsymbol{\Sigma}_{\mathbf{p}'}$

```
function [x_p, y_p, theta_p] = MAJEtatOdometrie(x, y, theta, delta_sg, delta_sd, L)

delta_s = (delta_sg + delta_sd)/2;
delta_theta = (delta_sd - delta_sg)/L;

x_p = x + delta_s*cos(theta + delta_theta/2); %% Mise à jour sur x
y_p = y + delta_s*sin(theta + delta_theta/2); %% Mise à jour sur y
theta_p = theta + delta_theta;                %% Mise à jour sur l'orientation
```

```
function SIGMApp = propageErreurs(SIGMAp, delta_sg, delta_sd, theta, L, kg, kd)

delta_s = (delta_sd + delta_sg)/2;
delta_theta = (delta_sd - delta_sg)/L;

SIGMAdelta = [kd*abs(delta_sd)          0;
                    0          kg*abs(delta_sg)];

nabla_f_p = [1  0 -delta_s*sin(theta + delta_theta/2);
             0  1  delta_s*cos(theta + delta_theta/2);
             0  0                    1                ];

nabla_f_Ddg11 = 0.5*cos(theta + delta_theta/2) - (delta_s/(2*L))*sin(theta + delta_theta/2);
nabla_f_Ddg12 = 0.5*cos(theta + delta_theta/2) + (delta_s/(2*L))*sin(theta + delta_theta/2);
nabla_f_Ddg21 = 0.5*sin(theta + delta_theta/2) + (delta_s/(2*L))*cos(theta + delta_theta/2);
nabla_f_Ddg22 = 0.5*sin(theta + delta_theta/2) - (delta_s/(2*L))*cos(theta + delta_theta/2);
nabla_f_Ddg31 = 1/L;
nabla_f_Ddg32 = -1/L;

nabla_f_Ddg = [nabla_f_Ddg11, nabla_f_Ddg12;
               nabla_f_Ddg21, nabla_f_Ddg22;
               nabla_f_Ddg31, nabla_f_Ddg32];

SIGMApp = nabla_f_p*SIGMAp*nabla_f_p' + nabla_f_Ddg*SIGMAdelta*nabla_f_Ddg';
```

① Prédiction

Exercice 1

```
function [x, y, theta, P] = prediction(xi, yi, thetai, Pi, delta_sg, delta_sd, L, kg, kd)

delta_s = (delta_sg + delta_sd)/2;
delta_theta = (delta_sd - delta_sg)/L;

theta = thetai + delta_theta;
if theta < 0
   theta = theta + 2*pi;
end
x = xi + delta_s*cos(thetai + delta_theta/2);
y = yi + delta_s*sin(thetai + delta_theta/2);

Q = [kd*abs(delta_sd)       0;
            0         kg*abs(delta_sg)];

Fx = [ 1  0 -delta_s*sin(thetai + delta_theta/2);
       0  1  delta_s*cos(thetai + delta_theta/2);
       0  0                    1                 ];

Fu = [ 0.5*cos(thetai + delta_theta/2) - (delta_s/2*L)*sin(thetai + delta_theta/2), ...
               0.5*cos(thetai + delta_theta/2) + (delta_s/(2*L))*sin(thetai + delta_theta/2);

       0.5*sin(thetai + delta_theta/2) + (delta_s/2*L)*cos(thetai + delta_theta/2), ...
               0.5*sin(thetai + delta_theta/2) - (delta_s/(2*L))*cos(thetai + delta_theta/2)

       1/L, -1/L ];

P = Fx*Pi*Fx' + Fu*Q*Fu';
```

Localisation d’un robot par EKF

① Prédiction

② **Correction**

a) Observation des primitives par mesure capteurs

b) Prédiction de mesure des primitives

c) Appariement des mesures

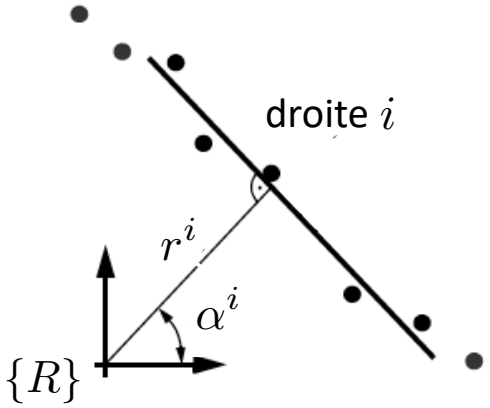
d) Estimation

② Correction

a) Observation des primitives par mesure capteurs

- Primitives de type droite
  - Une droite est paramétrée en coordonnées polaires:

$$\mathbf{z}^i = \begin{bmatrix} \alpha^i \\ r^i \end{bmatrix}^R, \quad i \in \{1, 2, \ldots, n\}$$



- n observations : $\mathbf{z} = [(\mathbf{z}^1)^T, \ldots, (\mathbf{z}^n)^T]^T \in \mathbb{R}^{2n}$
- Mesures dans le repère local $\{R\}$ du robot

② Correction

a) Observation des primitives par mesure capteurs

- n droites observées, $2n$ paramètres
- L'incertitude sur les droites est representée par la matrice de covariance :

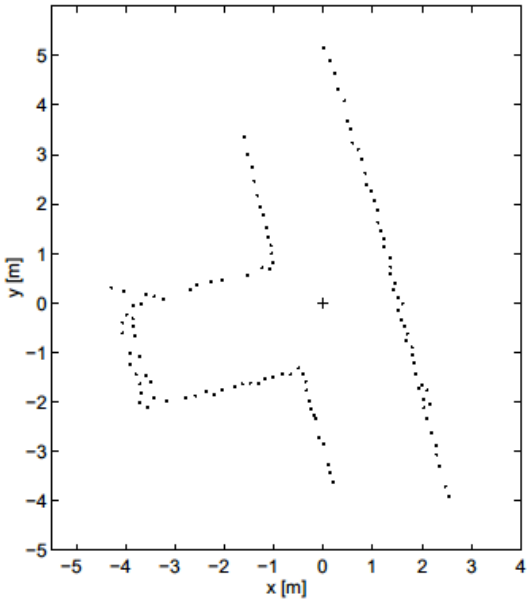
$$\mathbf{R}_t^i = \begin{bmatrix} \sigma_{\alpha\alpha}^i(t) & \sigma_{\alpha r}^i(t) \\ \sigma_{r\alpha}^i(t) & \sigma_{rr}^i(t) \end{bmatrix}, \quad i \in \{1, 2, \ldots, n\}$$

- Cette matrice de covariance peut être calculée à partir des incertitudes des mesures contribuant à chaque droite
- Dans notre implémentation Matlab, on choisira simplement:

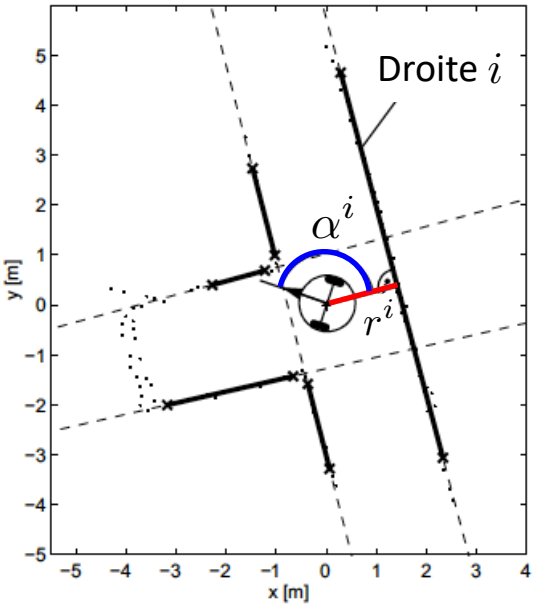
$$\mathbf{R}_t^i = \begin{bmatrix} 2.5\,\pi/180 & 0 \\ 0 & 10 \end{bmatrix}, \quad i \in \{1, 2, \ldots, n\}$$

② Correction

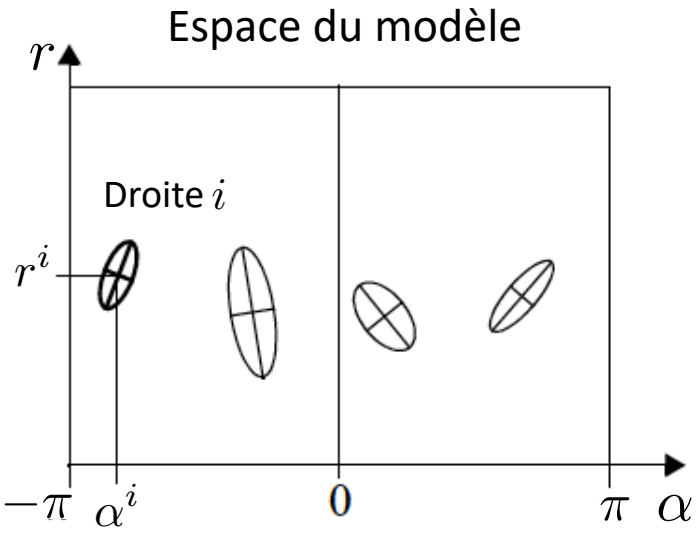
a) Observation des primitives par mesure capteurs



Données brutes générées par le télémètre laser 2D



Les droites sont extraites (par ex. avec RANSAC)



Représentation des paramètres des droites et des incertitudes associées, dans l'espace du modèle

Localisation d’un robot par EKF

① Prédiction

② **Correction**

a) Observation des primitives par mesure capteurs

b) Prédiction de mesure des primitives

c) Appariement des mesures

d) Estimation

② Correction

b) Prédiction de mesure des primitives

- **Utilise:**
  - La position prédite du robot $\widehat{\mathbf{x}}_t$ (calculée dans l'étape 1)
  - Le modèle de la primitive $\mathbf{m}^j$ (une droite):

$$\mathbf{m}^j = \begin{bmatrix} \alpha^j \\ r^j \end{bmatrix}^W$$

- **Donne:**
  - L'observation prédite pour chaque primitive

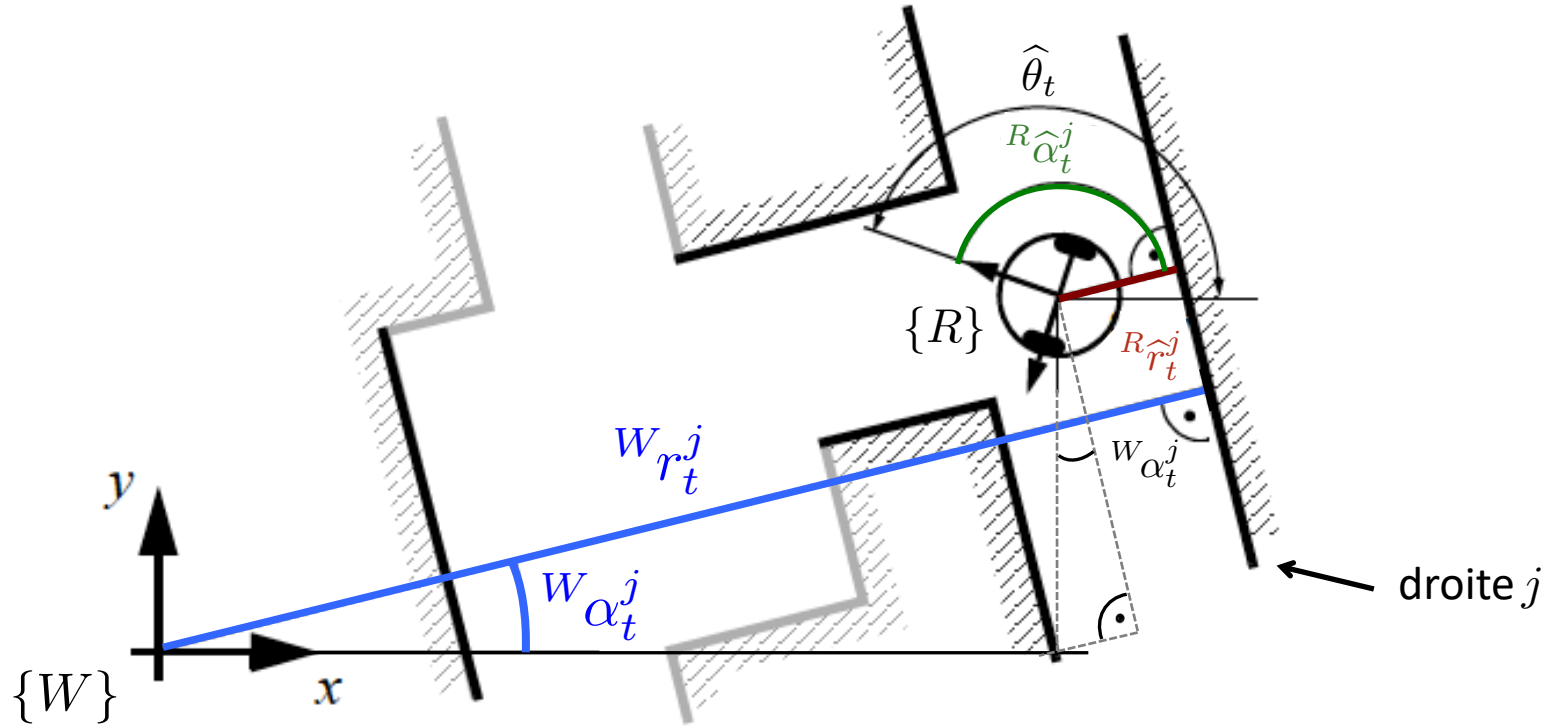
$$\widehat{\mathbf{z}}_t^j = \mathbf{h}^j(\widehat{\mathbf{x}}_t, \mathbf{m}^j)$$

dans le *repère local* $\{R\}$ du robot

② Correction

b) Prédiction de mesure des primitives

$$\widehat{\mathbf{z}}_t^j = {}^{R}\begin{bmatrix} \widehat{\alpha}_t^j \\ \widehat{r}_t^j \end{bmatrix} = \mathbf{h}^j(\widehat{\mathbf{x}}_t, \mathbf{m}^j) = \begin{bmatrix} {}^{W}\alpha_t^j - \widehat{\theta}_t \\ {}^{W}r_t^j - \widehat{x}_t \cos({}^{W}\alpha_t^j) - \widehat{y}_t \sin({}^{W}\alpha_t^j) \end{bmatrix}$$



② Correction

b) Prédiction de mesure des primitives

Par la suite on va utiliser la matrice jacobienne de $\mathbf{h}^j(\widehat{\mathbf{x}}_t,\ \mathbf{m}^j)$ par rapport à $\widehat{\mathbf{x}}_t$:

$$\mathbf{H}^j = \frac{\partial\, \mathbf{h}^j(\widehat{\mathbf{x}}_t,\ \mathbf{m}^j)}{\partial\, \widehat{\mathbf{x}}_t} = \begin{bmatrix} 0 & 0 & -1 \\ -\cos({}^W\alpha_t^j) & -\sin({}^W\alpha_t^j) & 0 \end{bmatrix}$$

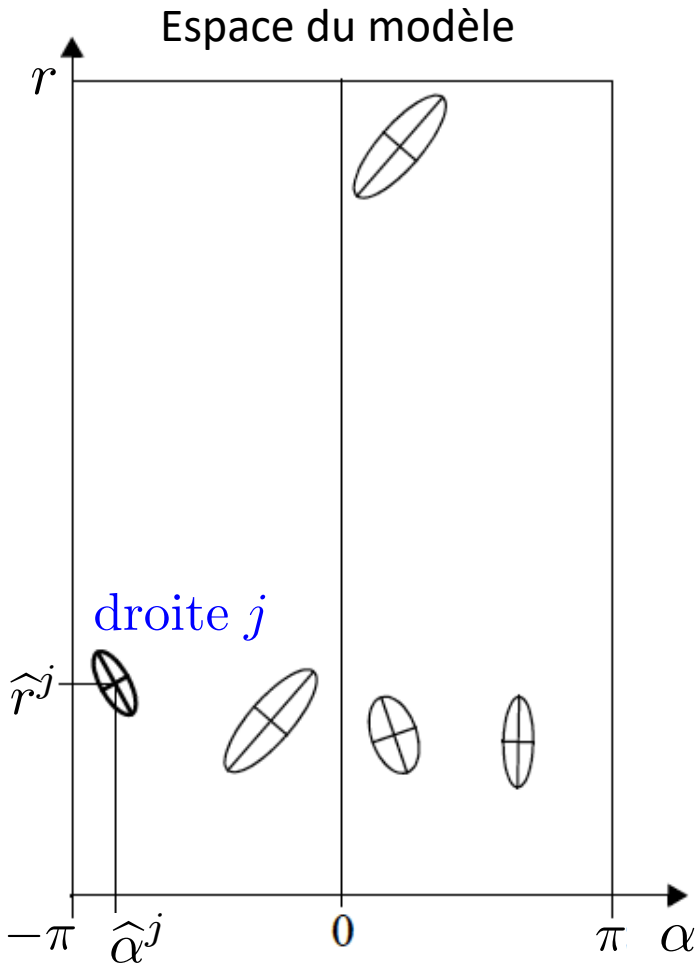
pour écrire les équations du filtre de Kalman

② Correction

b) Prédiction de mesure des primitives

- Les *prédictions de mesure* peuvent être représentées dans l'espace du modèle d'une façon similaire aux observations

- Le résultat de la prédiction de mesure sont des droites représentées dans le repère du robot $\{R\}$
  - Elles sont *incertaines*, parce que la prédiction de pose du robot $\widehat{\mathbf{x}}_t$ est incertaine



② Correction

b) Prédiction de mesure des primitives

$$\widehat{\mathbf{z}}_t^j = \overset{R}{\begin{bmatrix} \widehat{\alpha}_t^j \\ \widehat{r}_t^j \end{bmatrix}} = \mathbf{h}^j(\widehat{\mathbf{x}}_t, \mathbf{m}^j) = \begin{bmatrix} {}^W\alpha_t^j - \widehat{\theta}_t \\ {}^W r_t^j - \widehat{x}_t \cos({}^W\alpha_t^j) - \widehat{y}_t \sin({}^W\alpha_t^j) \end{bmatrix}$$

Exercice 2.2 (voir “initEnviro.m”, “genereTrajs.m” pour la gestion des droites)

$$\widehat{\mathbf{z}}_t = \text{predictionDeMesure}({}^W\alpha_t, {}^W r_t, \widehat{\mathbf{x}}_t)$$

```
function zPredit = predictionDeMesure(a_d, r_d, xi, yi, thetai)

for j = 1:length(a_d)
    zPredit(1, j) = a_d(j) - thetai;
    if (zPredit(1, j) < 0)
        zPredit(1, j) = zPredit(1, j) + 2*pi;
    end
    zPredit(2, j) = r_d(j) - xi*cos(a_d(j)) - yi*sin(a_d(j));
end
```

Localisation d’un robot par EKF

① Prédiction

② **Correction**

a) Observation des primitives par mesure capteurs

b) Prédiction de mesure des primitives

c) Appariement des mesures

d) Estimation

② Correction

c) Appariement des mesures

$$\mathbf{z}^i \overset{R}{=} \begin{bmatrix} \alpha^i \\ r^i \end{bmatrix} \quad \overset{?}{\Leftrightarrow} \quad \hat{\mathbf{z}}^j \overset{R}{=} \begin{bmatrix} \hat{\alpha}^j \\ \hat{r}^j \end{bmatrix}$$

- Trouver la meilleur correspondance entre:
  - Les mesures courantes (« indice i »)
  - Les prédictions de mesure (« indice j »)

- Pour chaque appariement (i, j) on calcule:
  - L'innovation
  - La matrice de covariance de l'innovation

② Correction

c) Appariement des mesures

- Pour chaque appariement $(i,\, j)$:
  - Calculer l'innovation:

$$\mathbf{v}_t^{ij} = \mathbf{z}_t^i - \mathbf{h}^j(\widehat{\mathbf{x}}_t,\, \mathbf{m}^j) = \begin{bmatrix} \alpha_t^i \\ r_t^i \end{bmatrix} - \begin{bmatrix} {}^W\alpha_t^j - \widehat{\theta}_t \\ {}^W r_t^j - \widehat{x}_t \cos({}^W\alpha_t^j) - \widehat{y}_t \sin({}^W\alpha_t^j) \end{bmatrix}$$

  - Calculer la covariance de l'innovation:

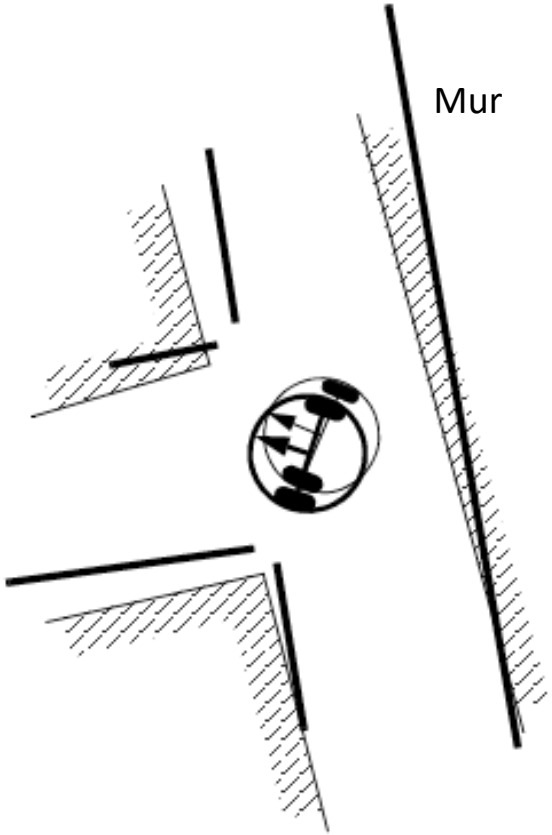
$$\mathbf{\Sigma}_{\mathrm{IN}_t}^{ij} = \mathbf{H}^j\, \widehat{\mathbf{P}}_t\, (\mathbf{H}^j)^T + \mathbf{R}_t^i$$

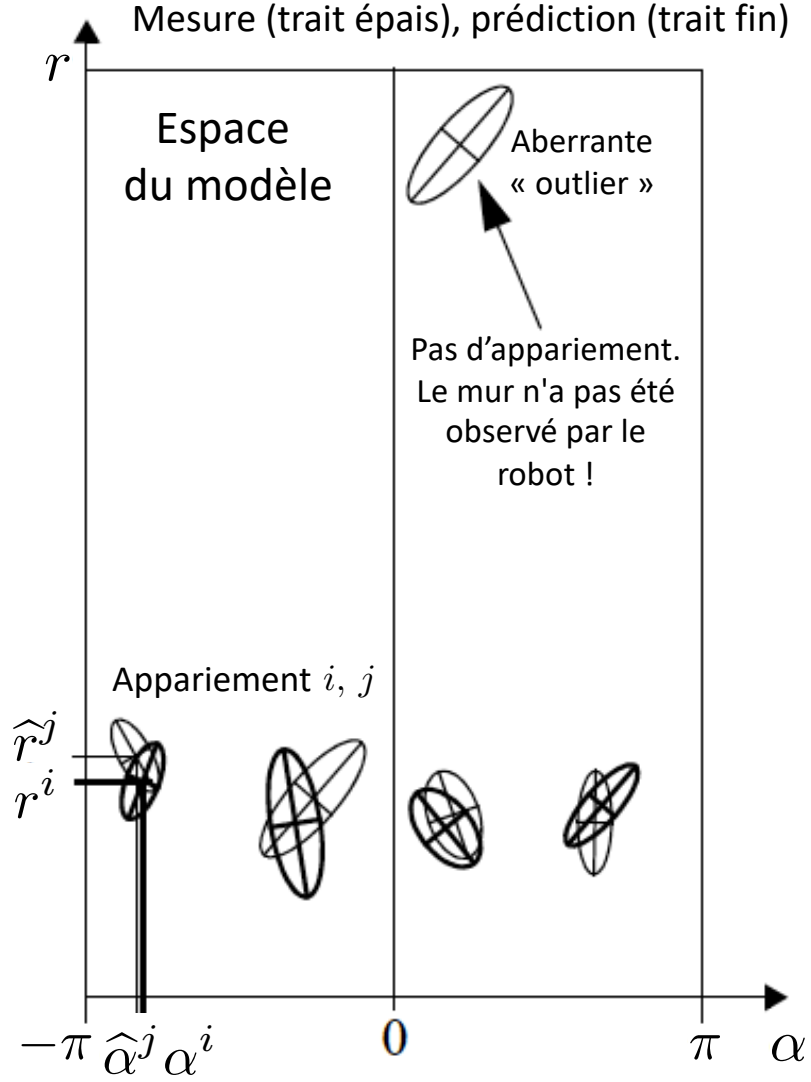
  - Pour valider l'appariement, on utilise la *distance de Mahalanobis* (au carré) avec un seuil $g > 0$:

$$(\mathbf{v}_t^{ij})^T (\mathbf{\Sigma}_{\mathrm{IN}_t}^{ij})^{-1}\, \mathbf{v}_t^{ij} \leq g^2$$

② Correction

c) Appariement des mesures





On cherche les meilleurs appariements.
Les primitives non appariées *sont rejetées*

② Correction

c) Appariement des mesures

- Trouver la correspondance entre les mesures $\mathbf{z}_t$ et les observations prédites $\widehat{\mathbf{z}}_t$
- Calculer l'innovation composite $\mathbf{v}_t$
- Calculer la covariance de l'innovation composite $\mathbf{\Sigma}_{\mathrm{IN}_t}$
- Empiler les $\mathbf{H}^j$ en $\mathbf{H}_t$

$$[\mathbf{v}_t,\, \mathbf{H}_t,\, \mathbf{\Sigma}_{\mathrm{IN}_t}] = \mathrm{appariement}(\widehat{\mathbf{z}}_t,\, \mathbf{z}_t,\, \widehat{\mathbf{P}}_t,\, \mathbf{R}_t,\, g)$$

Exercice 2.3

② Correction

c) Appariement des mesures

Exercice 2.3

```
function [v, H, SIGMAin] = appariement(zPredit, z, Pi, R, g)

for j=1:size(z,2)
    dM_max = 1e10;
    dM = dM_max;
    i = 1;
    while(i <= size(zPredit,2)) && (dM > g^2)
        % Matrice jacobienne de h^j
        Hj = [      0                   0              -1;
           -cos(zPredit(1,j))  -sin(zPredit(1,j))      0];

        % Innovation pour l'appariement candidat
        vij = z(:,i) - zPredit(:,j);

        % Calcul de la distance de Mahalanobis (au carré)
        SIGMAinij = Hj*Pi*Hj' + R(1+2*(i-1):1+2*(i-1)+1,1+2*(i-1):1+2*(i-1)+1);
        dM = vij' * inv(SIGMAinij) * vij;

        % Contribution a l'innovation et au jacobien composites
        if(dM < dM_max)
            dM_max = dM;
            v(1+2*(j-1):1+2*(j-1)+1) = vij;
            H(1+2*(j-1):1+2*(j-1)+1, :) = Hj;
        end

        i = i + 1;
    end
end
SIGMAin = H*Pi*H' + R; % Calcul de la covariance de l'innovation composite
v = v';
```

Localisation d’un robot par EKF

① Prédiction

② **Correction**

a) Observation des primitives par mesure capteurs

b) Prédiction de mesure des primitives

c) Appariement des mesures

d) Estimation

② Correction

d) Estimation

En se basant sur:

- La pose prédite $\widehat{\mathbf{x}}_t$ et la covariance $\widehat{\mathbf{P}}_t$
- Les mesures appariées

(I) Mise à jour de la pose: $\mathbf{x}_t = \widehat{\mathbf{x}}_t + \mathbf{K}_t \, \mathbf{v}_t$

- Gain de l'EKF:

$$\mathbf{K}_t = \widehat{\mathbf{P}}_t \, \mathbf{H}_t^T (\mathbf{\Sigma}_{\mathrm{IN}_t})^{-1}$$

(II) Mise à jour de la covariance: $\mathbf{P}_t = \widehat{\mathbf{P}}_t - \mathbf{K}_t \, \mathbf{\Sigma}_{\mathrm{IN}_t} \mathbf{K}_t^T$

$$[\mathbf{x}_t, \, \mathbf{P}_t] = \mathrm{estimation}(\widehat{\mathbf{x}}_t, \, \widehat{\mathbf{P}}_t, \, \mathbf{v}_t, \, \mathbf{H}_t, \, \mathbf{\Sigma}_{\mathrm{IN}_t})$$ **Exercice 2.4**

② Correction

Exercice 2.4

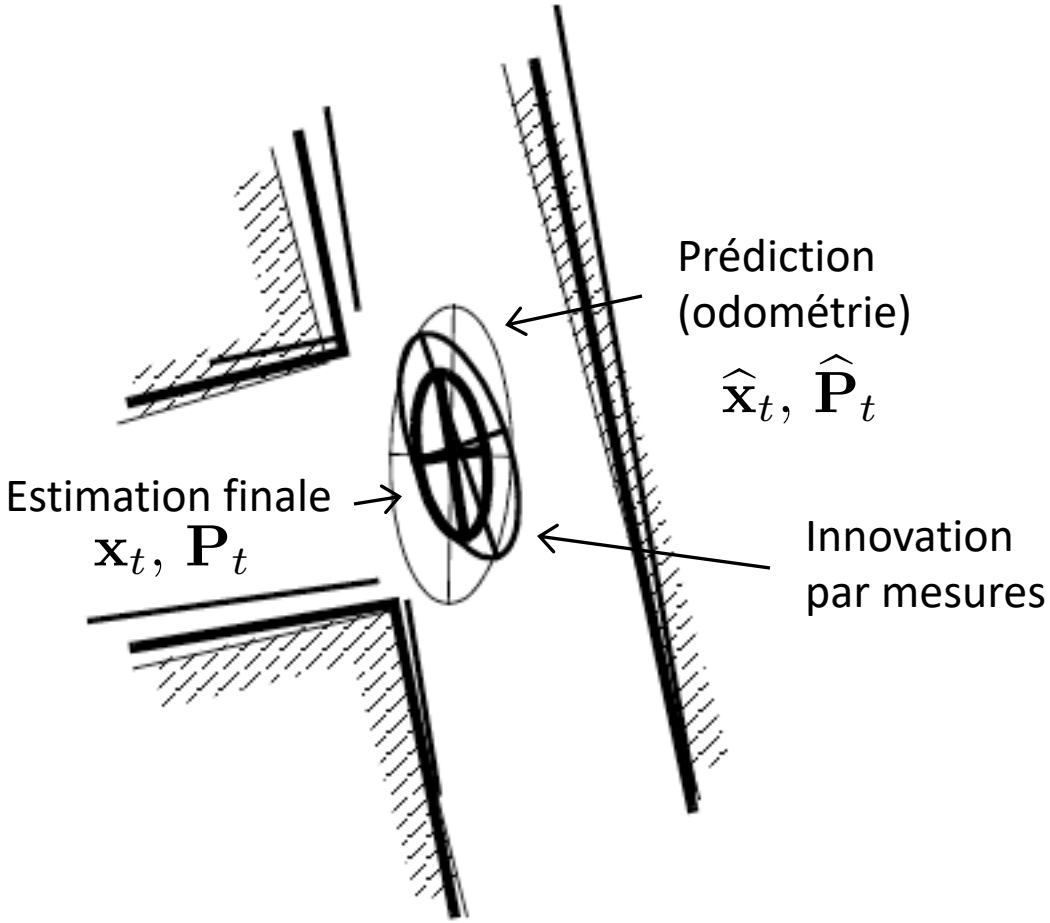
```
function [x, y, theta, P] = estimation(xi, yi, thetai, Pi, v, H, SIGMAin)

% Calcul du gain de Kalman
K = Pi * H' * inv(SIGMAin);

% Estimation de l'état
X = [xi, yi, thetai]' + K * v;

% Estimation de l'incertitude de l'état
P = Pi - K * SIGMAin * K';

% Mise en forme
x = X(1);
y = X(2);
theta = X(3);

if(theta < 0)
    theta = theta + 2*pi;
end
```

② Correction

Exercice 2.1: $[\mathbf{x}_t, \mathbf{P}_t] = \text{correction}(\widehat{\mathbf{x}}_t, \widehat{\mathbf{P}}_t, {}^{W}\alpha_t, {}^{W}r_t, {}^{R}\alpha_t, {}^{R}r_t, g)$

```
function [x, y, theta, P] = correction(xi, yi, thetai, Pi, a_d, r_d, a_d_o, r_d_o, g)

% alpha_droites -> a_d
% r_droites -> r_d
% alpha_droites_obs -> a_d_o
% r_droites_obs -> r_d_o

% Observation
z = [a_d_o ; r_d_o];
R = zeros(2*length(a_d_o),2*length(a_d_o));

% Bruit sur la perception extéroceptive. Variances sur la diagonale de la
% matrice R^j_t: sigma_aa = 2.5*pi/180, sigma_rr = 10

for i=1:length(a_d_o)
    R(1+2*(i-1):1+2*(i-1)+1, 1+2*(i-1):1+2*(i-1)+1) = [2.5*pi/180 0; 0 10];
end

% Prédiction de mesure
zPredit = predictionDeMesure(a_d, r_d, xi, yi, thetai);

% Appariement
[v, H, SIGMAin] = appariement(zPredit, z, Pi, R, g);

% Estimation
[x, y, theta, P] = estimation(xi, yi, thetai, Pi, v, H, SIGMAin);
```

② Correction



En fusionnant la *prédiction* de la pose du robot (trait fin) avec l'*innovation* obtenue par les mesures (trait épais), nous obtenons l'*estimation finale* de la pose du robot (trait très épais)

Sommaire des équations de l'EKF pour la localisation d'un robot

Prédiction

$$\widehat{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t)$$

$$\widehat{\mathbf{P}}_t = \mathbf{F}_\mathbf{x}\,\mathbf{P}_{t-1}\,\mathbf{F}_\mathbf{x}^T + \mathbf{F}_\mathbf{u}\,\mathbf{Q}_t\,\mathbf{F}_\mathbf{u}^T$$

Correction

$$\mathbf{x}_t = \widehat{\mathbf{x}}_t + \mathbf{K}_t\,\mathbf{v}_t$$

$$\mathbf{P}_t = \widehat{\mathbf{P}}_t - \mathbf{K}_t\,\boldsymbol{\Sigma}_{\mathrm{IN}_t}\mathbf{K}_t^T = \left(\widehat{\mathbf{P}}_t^{-1} + \mathbf{H}_t^T\,\mathbf{R}_t^{-1}\,\mathbf{H}_t\right)^{-1}$$

où $\mathbf{K}_t = \widehat{\mathbf{P}}_t\,\mathbf{H}_t^T(\boldsymbol{\Sigma}_{\mathrm{IN}_t})^{-1}$ est le gain de Kalman

Avantage

- Efficacité computationnelle (mécanisme *récursif*)

Inconvénient

- Si l'incertitude sur la pose du robot devient trop importante (par ex. suite à la collision avec un obstacle), l'EKF peut échouer et l'erreur d'estimation peut diverger