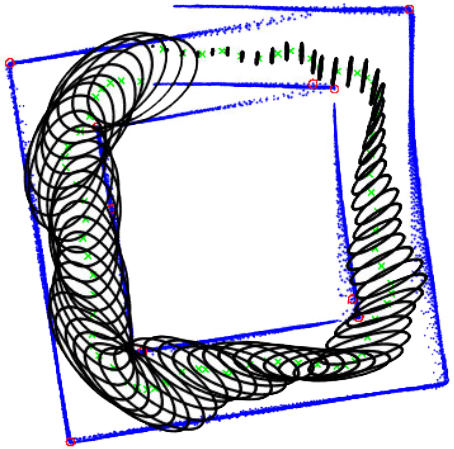
UNIVERSITÉ de Picardie Jules Verne

Localisation et navigation de robots

TD1 : Odométrie

Fabio MORBIDI



2023-2024

TD1: Odométrie

Objectif: Estimation de la trajectoire d'un robot mobile

- Calculer la pose (x, y, θ) d'un robot unicycle (à conduite différentielle) à partir de l'odométrie
- Tracer la position instantanée et simuler la trajectoire du robot
- Identification de l'erreur (incertitude) par le calcul de la matrice de covariance
- Extraction et affichage des ellipses d'incertitude sur la trajectoire

TD1: Odométrie

<u>Matériel</u>: Répertoire *packTD1.zip*

Fichier « simu.m » utilise :

- Fichier « initRobot » pour les paramètres du robot
- Fichier « initBruits » pour simuler:
  - Le bruit d'une commande mal appliquée
  - Le bruit de perception des capteurs
- Fichier « genereTrajs »
  - Pour générer la trajectoire du robot:
    - S'il applique *parfaitement* la commande
    - Si on prends en compte les bruits
  - Donne les valeurs des angles de chaque roue: φ_g et φ_d

TD1: Odométrie

Matériel: **Répertoire** *packTD1.zip*

Répertoire « outils » contient des fonctions:

- « Drawarrow » : Dessiner une flèche
- « Drawroundedrect » : Dessiner un rectangle
- « Drawellipse » : Dessiner une ellipse
- « Drawrobot » : Dessiner un robot

Exercice 1

Fonction « MAJEtatOdometrie »

- **Entrées :**
  - Les distances parcourues par la roue gauche et droite entre deux instants de temps et la pose courante $\mathbf{p}$ du robot
  - La largeur de l'essieu L
- **Sortie :**
  - Mise à jour de la pose : $\mathbf{p}' = [x', y', \theta']^T$

Utiliser cette fonction dans le fichier « simu.m » pour tracer le robot à chaque instant

Exercice 1

Fonction « MAJEtatOdometrie »

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \qquad \Delta s = \frac{\Delta s_d + \Delta s_g}{2}, \qquad \Delta x = \Delta s \cos(\theta + \Delta\theta/2)$$

$$\Delta\theta = \frac{\Delta s_d - \Delta s_g}{L}, \qquad \Delta y = \Delta s \sin(\theta + \Delta\theta/2)$$

$$\mathbf{p}' = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \mathbf{p} + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta/2) \\ \Delta s \sin(\theta + \Delta\theta/2) \\ \Delta\theta \end{bmatrix}$$

Exercice 1

Fonction « MAJEtatOdometrie »

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \qquad \Delta s = \frac{\Delta s_d + \Delta s_g}{2}, \qquad \Delta x = \Delta s \cos(\theta + \Delta\theta/2)$$

$$\Delta\theta = \frac{\Delta s_d - \Delta s_g}{L}, \qquad \Delta y = \Delta s \sin(\theta + \Delta\theta/2)$$

$$\mathbf{p}' = \mathbf{f}(x, y, \theta, \Delta s_d, \Delta s_g) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_d + \Delta s_g}{2} \cos\left(\theta + \frac{\Delta s_d - \Delta s_g}{2L}\right) \\ \frac{\Delta s_d + \Delta s_g}{2} \sin\left(\theta + \frac{\Delta s_d - \Delta s_g}{2L}\right) \\ \frac{\Delta s_d - \Delta s_g}{L} \end{bmatrix}$$

Exercice 1

Fonction « MAJEtatOdometrie »

```
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 theta
```

Exercice 1

Dans le ficher « simu.m »

```
clc
clear all
close all

addpath('outils/')

% nbIter est le nombre d'instants de la simulation
nbIter = 100;

% Figure pour afficher tous les resultats
figure(1)
clf
hold on

% Initialisation des scripts
initRobot;          % pour les dimensions du robot (r est le rayon d'une roue
                    % et L est la longueur de l'essieu)
initBruits;         % pour les parametres des bruits aléatoires de la commande
                    % et des capteurs

genereTrajs;
```

Exercice 1

```
% Initialisation de la position du robot
xOdom(1) = 0.0;
yOdom(1) = 0.0;
thetaOdom(1) = 0.0;

for i=1:nbIter

    diff_phig = phig(i+1)-phig(i);
    diff_phid = -(phid(i+1)-phid(i));

    delta_sg = diff_phig*r;
    delta_sd = diff_phid*r;
    %%  Exercice 1 %%
    [xOdom(i+1), yOdom(i+1), thetaOdom(i+1)] = ...
        MAJEtatOdometrie(xOdom(i), yOdom(i), thetaOdom(i),delta_sg, delta_sd,L );

    % Affiche les positions successives du robot
     drawrobot([xOdom(i) yOdom(i) thetaOdom(i)], 'k', 4, 250, 300);
     plot(xOdom,yOdom, 'r', 'LineWidth', 3)
     pause(0.02)
end
hold off
axis equal
title('Trajectoires')
h = legend(' Application parfaite de la commande',...
           ' Réelle (commande bruitée)',...
           ' Obtenue par odométrie','Location','NorthWest');
xlabel('x [m]')
ylabel('y [m]')
```

Exercice 2

Propagation de l'erreur:

Fonction « *propageErreurs* » pour le calcul de la matrice de covariance $\mathbf{\Sigma}_{\mathbf{p}'}$

- **Entrées :**
  - La matrice de covariance $\mathbf{\Sigma}_{\mathbf{p}}$ associée à l'état précédent
  - L'incrément courant
  - L'état à l'instant précédent
  - Les paramètres L, k_g, k_d
- **Sortie :**
  - La matrice de covariance $\mathbf{\Sigma}_{\mathbf{p}'}$

Exercice 2

$$\boxed{\mathbf{\Sigma}_{\mathbf{p}'} = \nabla_{\mathbf{p}}\mathbf{f} \cdot \mathbf{\Sigma}_{\mathbf{p}} \cdot (\nabla_{\mathbf{p}}\mathbf{f})^T + \nabla_{\mathbf{\Delta}_{dg}}\mathbf{f} \cdot \mathbf{\Sigma}_{\Delta} \cdot (\nabla_{\mathbf{\Delta}_{dg}}\mathbf{f})^T}$$

$$\mathbf{\Sigma}_{\Delta} = \mathrm{cov}(\Delta s_d, \Delta s_g) = \begin{bmatrix} k_d\,|\Delta s_d| & 0 \\ 0 & k_g\,|\Delta s_g| \end{bmatrix}$$

$$\nabla_{\mathbf{p}}\mathbf{f} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x} & \frac{\partial \mathbf{f}}{\partial y} & \frac{\partial \mathbf{f}}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta s \sin(\theta + \Delta\theta/2) \\ 0 & 1 & \Delta s \cos(\theta + \Delta\theta/2) \\ 0 & 0 & 1 \end{bmatrix}$$

$$\nabla_{\mathbf{\Delta}_{dg}}\mathbf{f} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \Delta s_d} & \frac{\partial \mathbf{f}}{\partial \Delta s_g} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\cos(\theta + \Delta\theta/2) - \frac{\Delta s}{2L}\sin(\theta + \Delta\theta/2) & \frac{1}{2}\cos(\theta + \Delta\theta/2) + \frac{\Delta s}{2L}\sin(\theta + \Delta\theta/2) \\ \frac{1}{2}\sin(\theta + \Delta\theta/2) + \frac{\Delta s}{2L}\cos(\theta + \Delta\theta/2) & \frac{1}{2}\sin(\theta + \Delta\theta/2) - \frac{\Delta s}{2L}\cos(\theta + \Delta\theta/2) \\ 1/L & -1/L \end{bmatrix}$$

$$\Delta s = \frac{\Delta s_d + \Delta s_g}{2}, \quad \Delta\theta = \frac{\Delta s_d - \Delta s_g}{L}$$

Exercice 2

Propagation de l'erreur:

Fonction « *propageErreurs* » pour le calcul de la matrice de covariance $\boldsymbol{\Sigma}_{\mathbf{p}'}$

```
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';
```

Exercice 2

Propagation de l'erreur:

Fonction « *propageErreurs* » pour le calcul de la matrice de covariance $\boldsymbol{\Sigma}_{\mathbf{p}'}$

Dans le ficher « simu.m »

```
% Matrice de covariance initiale
SIGMAp = 1e-4*eye(3);
                :
                :
diff_phig = phig(i+1)-phig(i);
diff_phid = -(phid(i+1)-phid(i));

delta_sg = diff_phig*r;
delta_sd = diff_phid*r;
```

```
%% Exercice 1 %%
[xOdom(i+1), yOdom(i+1), thetaOdom(i+1)] = ...
MAJEtatOdometrie(xOdom(i), yOdom(i), thetaOdom(i), delta_sg, delta_sd, L);
```

```
%% Exercice 2 %%
SIGMAp = propageErreurs(SIGMAp, delta_sg, delta_sd, thetaOdom(i), L, kg  kd);
```

Exercice 3

Propagation de l'erreur:

Fonction « *extraitEllipse* » pour calculer l'ellipse d'incertitude associée à $\mathbf{\Sigma}_{\mathbf{p}'}$

Affichez ces ellipses (cf. dossier *outils*) à un pas régulier sur la trajectoire du robot (mais n'affichez pas toutes les ellipses, sinon ce sera illisible !)

Exercice 3

Propagation de l'erreur:

Fonction « *extraitEllipse* » pour calculer l'ellipse d'incertitude associée à $\boldsymbol{\Sigma}_{\mathbf{p}'}$

Remarque:

- Les racines carrées des valeurs propres les plus fortes de la matrice de covariance donnent les longueurs des demi-axes de l'ellipse
- Le vecteur propre associé à la valeur propre la plus forte nous donne des informations sur l'orientation de l'ellipse

Exercice 3

Propagation de l'erreur:

Fonction « *extraitEllipse* » pour calculer l'ellipse d'incertitude associée à $\boldsymbol{\Sigma}_{\mathbf{p}'}$

```
function [da1, da2, theta_e] = extraitEllipse(SIGMAp)

[V, D] = eigs(SIGMAp);

% Longeur des demi-axes (au carré) de l'ellipse d'incertitude

da1 = D(3,3);
da2 = D(2,2);

% Orientation de l'ellipse d'incertitude (radians)

theta_e = atan2(V(2,3), V(1,3));

if theta_e < 0
   theta_e = theta_e + pi;
end
```

Exercice 3

Propagation de l'erreur:

Fonction « *extraitEllipse* » pour calculer l'ellipse d'incertitude associée à $\boldsymbol{\Sigma}_{\mathbf{p}'}$

Dans le ficher « simu.m »

```
diff_phig = phig(i+1)-phig(i);
diff_phid = -(phid(i+1)-phid(i));

delta_sg = diff_phig*r;
delta_sd = diff_phid*r;
```

```
%% Exercice 1 %%
[xOdom(i+1), yOdom(i+1), thetaOdom(i+1)] = ...
MAJEtatOdometrie(xOdom(i), yOdom(i), thetaOdom(i), delta_sg, delta_sd, L);
```

```
%% Exercice 2 %%
SIGMAp = propageErreurs(SIGMAp, delta_sg, delta_sd, thetaOdom(i), L, kg, kd);
```

```
%% Exercice 3 %%
[da1(i+1), da2(i+1), theta_e(i+1)] = extraitEllipse(SIGMAp);

if mod(i,5) == 0
   drawellipse([xOdom(i+1), yOdom(i+1), theta_e(i+1)], 3*sqrt(da1(i+1)), 3*sqrt(da2(i+1)), 'k');
   plot(xOdom(i+1), yOdom(i+1),'k.','MarkerSize',14)
end
```