Rapport

Project SY15: Traversée d'un labyrinthe

Beijia MAO

Yiguang MEI

Yijian TIAN

Tâche 2.7.1
Définissez le modèle de prédiction du turtlebot.
— De quelles variables est composé l'état du système ?
— Les vitesses longitudinale et angulaire doivent-elles être dans l'état du véhicule ?
— Quelle dérivée de la position considérez-vous nulle ?
— Donnez les équations d'évolution de votre état, en temps discret.

l'état du système :

$$\chi = \begin{bmatrix} x \\ y \\ \theta \\ v \\ w \end{bmatrix}$$

Les vitesses longitudinales et angulaires doivent être incluses dans l'état du véhicule, car il est nécessaire d'utiliser les informations de vitesse pour estimer la position du véhicule. En même temps, ces vitesses peuvent être obtenues directement à partir des capteurs pour effectuer des corrections. Cela constitue un filtre de Kalman.

Puisque nous devons seulement contrôler le mouvement du véhicule sur un plan bidimensionnel, nous supposons que la coordonnée Z reste constante, c'est-à-dire que la dérivée de Z est nulle.

Voici les équations d'évolution de votre état, en temps discret

$$f_1 : x_{k+1|k} = x_{k|k} + v_{k|k} \cos(\theta_{k|k}) \cdot \Delta t$$

$$f_2 : y_{k+1|k} = y_{k|k} + v_{k|k} \sin(\theta_{k|k}) \cdot \Delta t$$

$$f_3 : \theta_{k+1|k} = \theta_{k|k} + w_{k|k} \cdot \Delta t$$

$$f_4 : v_{k+1|k} = v_{k|k}$$

$$f_5 : w_{k+1|k} = w_{k|k}$$

L'intervalle de temps est Δt

Tâche 2.7.2
Définissez le modèle de correction du turtlebot. Vous pourrez utiliser les données issues du robot : odométrie (vitesse longitudinale v et vitesse angulaire ωodom) et/ou IMU (accélération linéaire a et vitesse angulaire ωIMU).
— Quelle donnée issue du robot est directement compatible avec votre état, sans dériver ni intégrer ? (Si vous n'en trouvez aucune, il faudra peut-être revoir votre état)
— Donnez les équations du modèle de correction, permettant de prédire l'observation en fonction de l'état.

vitesse longitudinale v et vitesse angulaire ωodom sont directement compatibles avec l'état

les équations du modèle de correction :

$$
\begin{aligned}
x_{k|k} &= x_{k|k-1} \\
y_{k|k} &= y_{k|k-1} \\
\theta_{k|k} &= \theta_{k|k-1} \\
v_{k|k} &= v_{m,k} \\
w_{k|k} &= w_{m,k}
\end{aligned}
$$

Tâche 2.8.1
Donnez la Jacobienne F en fonction du modèle de prédiction défini précédemment.

$$
F = \begin{bmatrix}
\frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial \theta} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial w} \\
\frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial \theta} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial w} \\
\frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial \theta} & \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial w} \\
\frac{\partial f_4}{\partial x} & \frac{\partial f_4}{\partial y} & \frac{\partial f_4}{\partial \theta} & \frac{\partial f_4}{\partial v} & \frac{\partial f_4}{\partial w} \\
\frac{\partial f_5}{\partial x} & \frac{\partial f_5}{\partial y} & \frac{\partial f_5}{\partial \theta} & \frac{\partial f_5}{\partial v} & \frac{\partial f_5}{\partial w}
\end{bmatrix}
$$

$$
F = \begin{bmatrix}
1 & 0 & -v\sin(\theta)\Delta t & \cos(\theta)\Delta t & 0 \\
0 & 1 & v\cos(\theta)\Delta t & \sin(\theta)\Delta t & 0 \\
0 & 0 & 1 & 0 & \Delta t \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
$$

Tâche 2.8.2
Donnez la matrice C en fonction du modèle de correction défini précédemment.

$$\chi_{k|k} = \chi_{k|k-1} + K_k(Z - C\chi_{k|k-1})$$

$$P_{k|k} = (I - K_k C)P_{k|k-1}$$

$$K_k = P_{k|k-1}C^T(CP_{k|k-1}C^T + R_k)^{-1}$$

Z est le vecteur d'observation :

$$Z = [v\ w]^T$$

C est la matrice d'observation :

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

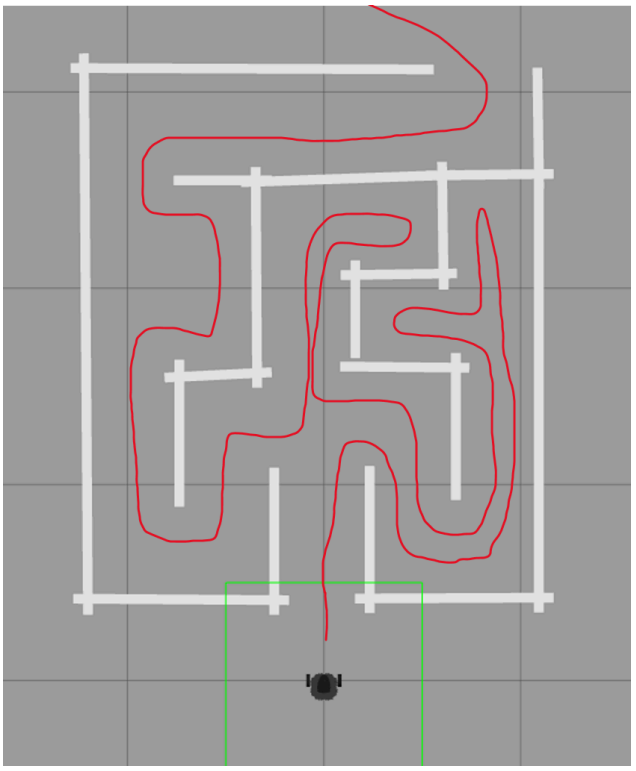
3.1 Objectif 1 : Atteindre la position demandée

Introduction:

Cette partie vise à présenter comment utiliser l'algorithme de recherche en profondeur pour faire sortir un robot d'un labyrinthe, en suivant en particulier la stratégie "toujours à droite".

Principe de l'algorithme

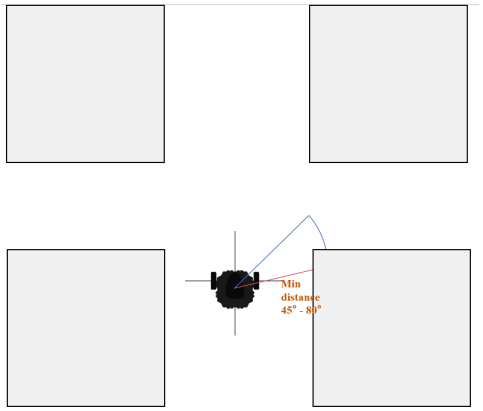
La recherche en profondeur est un algorithme qui explore un arbre ou un graphe en partant d'un nœud initial, en s'engageant dans une voie jusqu'à ce qu'il ne puisse plus continuer, puis en revenant en arrière pour essayer d'autres chemins.



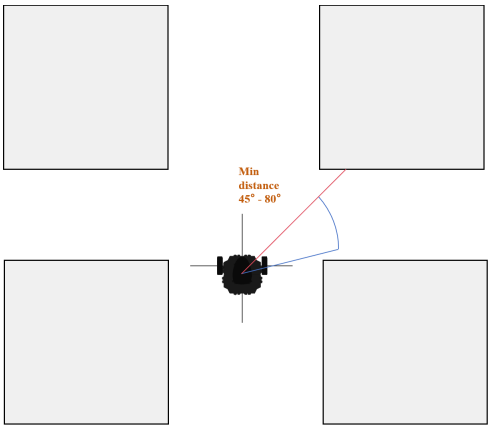
Méthode pour détecter s'il y a un chemin à droite

Pour déterminer si une intersection sur la droite a été détectée, on évalue si la distance la plus courte dans un angle spécifique sur la droite présente une variation significative.

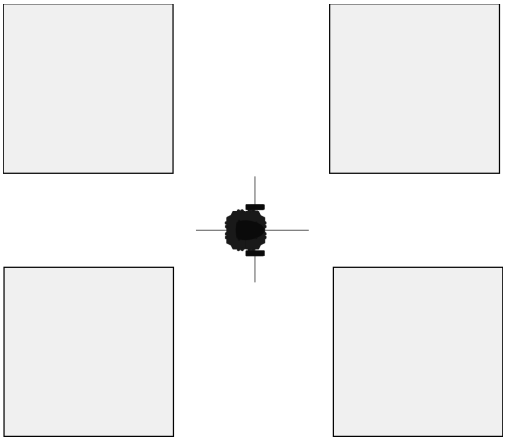
Le processus détaillé est illustré dans l'image suivante :



Avant de détecter l'intersection, la distance varie de manière relativement uniforme.



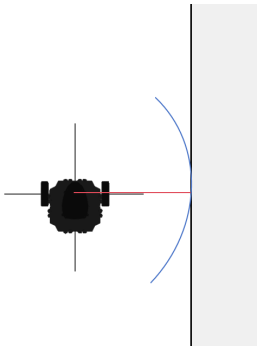
Lorsque l'intersection sur la droite est détectée, la distance la plus courte présente une variation brusque.



Après avoir détecté l'intersection, le robot se dirigera vers la zone centrale de l'intersection et effectuera une rotation de 90 degrés dans le sens horaire.

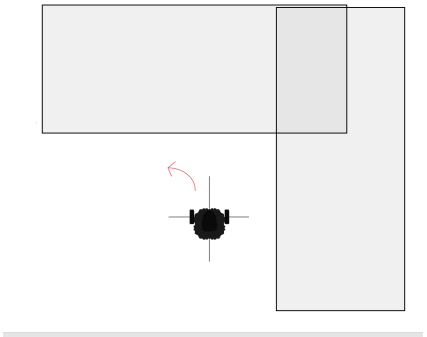
Direction correcte :

Pour assurer que le robot avance dans la bonne direction, nous veillons à ce que le côté droit du robot reste toujours perpendiculaire au mur et que le robot ajuste continuellement sa trajectoire en conséquence.



Une situation particulière :

Nous souhaitons que notre robot tourne toujours à droite, sauf dans un cas particulier. Lorsque le robot rencontre un angle mort comme illustré dans l'image, nous déterminerons si le robot a rencontré un obstacle et si la distance sur le côté gauche est nettement supérieure à la distance sur le côté droit. Dans ce cas, le robot tourne à gauche. Cette stratégie permet de garantir que, lorsque la sortie du labyrinthe se trouve sur le mur extérieur du labyrinthe, nous puissions toujours atteindre la sortie.



Sortie du labyrinthe :

Lorsque le champ de vision du robot devient dégagé dans un certain angle, nous déterminons que le robot a quitté le labyrinthe. Nous désactivons alors la logique liée au labyrinthe et utilisons les coordonnées obtenues par le filtre de Kalman pour diriger le robot vers les coordonnées et l'angle cibles.

3.2 Objectif 2 : Positionnement relatif à une cible (docking)

Le robot doit être arrêté à 20 cm du panneau, à la normale du panneau et face au panneau.

Nous atteignons cet objectif en suivant les étapes suivantes.

1. identifier l'emplacement exact du panneau à partir des données radar, y compris les informations concernant les deux extrémités du panneau.

Nous avons fixé un seuil d'intensité de réflexion et lu l'intensité de toutes les réflexions du radar, puis nous les avons filtrées jusqu'aux points mesurés sur le panneau. Les points les plus proches et les plus éloignés du robot sont alors considérés comme les deux extrémités.

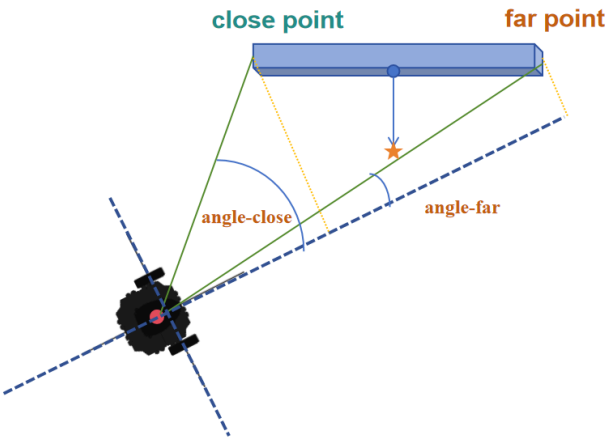
2. calculer les coordonnées de l'emplacement final du robot (dans le système de coordonnées mondiales) sur la base des informations de position du panneau.

```
far_x = distance_far * math.cos(angle_far)
far_y = distance_far * math.sin(angle_far)
close_x = distance_close * math.cos(angle_close)
close_y = distance_close * math.sin(angle_close)
```

Coordonnées de destination basées sur le calcul des vecteurs verticaux

```
def find_perpendicular_points_near_midpoint(x1, y1, x2, y2, distance_from_midpoint):
    mx, my = calculate_midpoint(x1, y1, x2, y2)
    distance = calculate_distance(x1, y1, x2, y2)
    ux, uy = calculate_unit_vector(x1, y1, x2, y2, distance)
    vx1, vy1 = -uy, ux
    vx2, vy2 = uy, -ux
    point1 = (mx + distance_from_midpoint * vx1, my + distance_from_midpoint * vy1)
    point2 = (mx + distance_from_midpoint * vx2, my + distance_from_midpoint * vy2)
    return point1, point2
```

3. commander le robot pour qu'il se rende à l'emplacement cible.



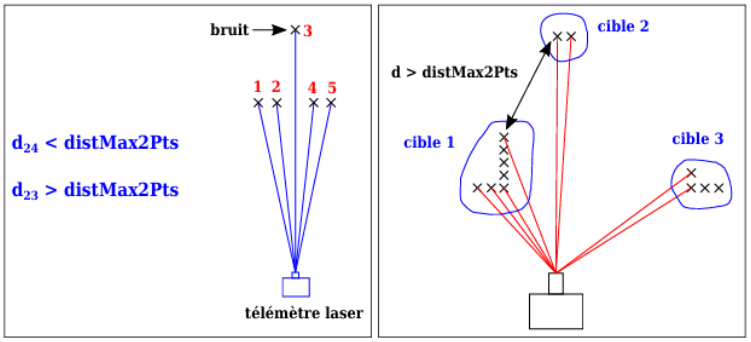
Nous calculons la position du point d'arrivée lorsque le robot démarre. L'avantage de cette méthode est que le système de coordonnées mondiales du robot au point de départ est (0,0) et que la direction du système de coordonnées du robot est la même que celle du système de coordonnées mondiales, de sorte que la position calculée du point cible se trouve dans les coordonnées du système de coordonnées mondiales.
De cette manière, bien que nous ne puissions pas détecter et mettre à jour la position du point cible en temps réel lorsque le robot avance, nous pouvons réduire le processus de changement de système de coordonnées et simplifier le calcul. Les résultats de nos tests de simulation sont également plus précis.

Mais nous avons rencontré un problème dans le processus réel : lorsque le radar du robot est situé au milieu du panneau de signalisation, le point le plus proche se trouve également au milieu du panneau, tandis que le point le plus éloigné se trouve aux extrémités du panneau, ce qui entraîne un point médian incorrect.

Par conséquent, nous avons modifié notre stratégie : nous obtenons des points de réflexion continus. Si l'intensité d'un point de réflexion augmente soudainement puis diminue, et que les points de réflexion continus suivants maintiennent une intensité élevée ou basse, nous considérons alors que le premier point de cette série de changements soudains est l'une des extrémités du panneau de signalisation. Ainsi, la voiture peut fonctionner dans toutes les positions.

4.Une meilleure façon

Bien que dans les simulations gazo, les chariots se soient garés parfaitement, dans les essais pratiques, les chances de succès sont faibles. Parce qu'en réalité, l'intensité de la réflexion interférait trop. Pour améliorer la précision, nous pouvons utiliser un algorithme de clustering.



On élimine dans un premier temps les données aberrantes, à savoir tout point situé entre deux points proches mais qui est loin de ces 2 points. Puis on classera les points selon la règle suivante : deux points consécutifs de distance inférieure à une valeur donnée(distMax2Pts) sont considérés provenant d'une même cible. La valeur distMax2Pts est calculée à partir des caractéristiques du télémètre.

De cette façon, on peut déterminer la position des panneaux de signalisation en estimant la longueur des cibles de clustering, ce qui réduit les effets des perturbations sonores.

Mais en fin de compte, nous n'avons pas réussi à implémenter cette méthode parce que lors du test du chariot, il y avait un problème avec le mode de passerelle de notre machine virtuelle et la communication entre le chariot. Nous avions juste le temps de tester l'algorithme simple précédent.