Projet long M1 Robot suiveur de ligne

Yani Amrioui Idir Lankri

30 mai 2017

Introduction

Problème

Programmer un robot capable d'aller d'un point de départ à un point d'arrivée en suivant une ligne

Exemples d'application

- Manutention : déplacer des charges lourdes, des marchandises dans un entrepôt ou une usine
- Aider des personnes à mobilité réduite dans leur vie de tous les jours (ex : apporter des objets)

Fonctionnalités

- Apprentissage de couleurs
- Reconnaissance de couleurs
- Suivi de lignes droites et courbes

Scénario typique d'utilisation

Contexte

On dispose d'un circuit (ligne tracée sur un certain fond) et on veut aller d'un point A à un point B de ce circuit en suivant ce circuit.

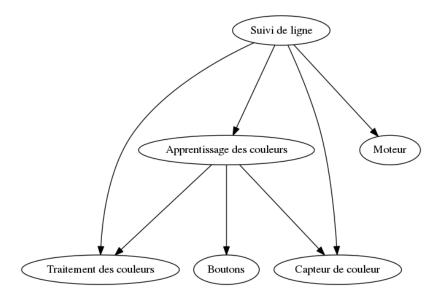
2 étapes

- 1. Apprendre au robot les couleurs du circuit
- 2. Placer le robot sur le circuit, puis le lancer afin qu'il suive la ligne

Décomposition du problème

- 1. Faire apprendre les couleurs au robot
- 2. Faire que le robot soit capable de reconnaître ces couleurs
- 3. Suivre une ligne en traitant les données obtenues via le capteur de couleur

Architecture du logiciel



Points techniques

Interaction avec le robot

Communication avec le robot via le système de fichiers (device files)

Compilation croisée

- Problème : générer du code machine exécutable par le robot sur une autre architecture
- ▶ Solution : Docker

Programmation

```
1(* [member col col cloud] returns true if and only if [col]
      belongs to \lceil co\overline{l} \ cloud1. *)
 3let member col col cloud =
 4 (* We suppose that colors composing a color cloud follows a Gaussian
       distribution. Thus, a random color of the cloud is between
       center - 3*sigma and center + 3*sigma with probability 99.7%. *)
 7 let delta = shift 3. col cloud.sigma in
   geg col (diff col cloud.center delta) &&
9 leg col (add col cloud.center delta)
10
11(* Return the cloud [c] among [col clouds] such that the distance
    between [c.center] and [col] is minimal. *)
13let nearest cloud col col clouds =
14 let aux col (old cloud, old dist) new cloud =
15
      let new dist = dist col (center new cloud) in
16
      if new dist < old dist then (new cloud, new dist) else (old cloud, old dist)
17
   in
18 match col clouds with
19 | [] -> assert false
20 | col cloud :: col clouds ->
21
      let dist = dist col (center col cloud) in
22
      fst (List.fold left (aux col) (col cloud, dist) col clouds)
23
24let recognize col known cols =
25 match List.filter (member col) known cols with
   (* There is no color recognized without doubt. In this case we return
       the "nearest" color among known colors. *)
28
    | [] -> Probable.Maybe (nearest cloud col known cols)
29
30
   (* We decide between the candidates by computing the "nearest"
31
       cloud. *)
32
    col clouds -> Probable.Sure (nearest cloud col col clouds)
```

Conclusion

Enseignements de ce projet

- Découverte de la robotique et de la programmation de système embarqué
- Difficulté de développer et tester un logiciel dépendant de capteurs

Améliorations possibles

- Utiliser un historique de navigation
- Gérer des circuits complexes : sélection de chemin par un code couleur