UNIVERSITATEA "POLITEHNICA" DIN TIMISOARA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL AUTOMATICĂ ŞI INFORMATICĂ
APLICATĂ

FOTBALIST

PROIECT SINCRETIC I

AUTORI: Dorin DUNCA
Cătălin ZORLESCU
Adrian-Ioan DEATCU

Coordonatori: Conf.dr.ing. Florin DRĂGAN,As. ing. Emil VOIŞAN

Cuprins

1. Introducere

Robotica mobilă este un domeniu dinamic și în continuă expansiune, având aplicații diverse în sectoare precum logistică, agricultură, medicină și explorare spațială. Roboții mobili sunt sisteme autonome sau semi-autonome capabile să navigheze în medii fizice, luând decizii în timp real pe baza datelor furnizate de senzori.

Un aspect esențial al roboticii mobile este conducerea autonomă sau asistată, întâlnită atât în vehicule autonome, cât și în roboți de dimensiuni reduse. Aceste sisteme complexe combină tehnologii avansate, precum procesarea imaginilor, detectarea obiectelor, navigația bazată pe senzori și controlul mișcării, pentru a opera eficient în medii variate și pentru a interacționa în siguranță cu mediul înconjurător.

2. Prezentarea temei – Fotbalist

Tema proiectului este dezvoltarea unui robot mobil autonom capabil să simuleze rolul unui fotbalist, cu abilități de a detecta o minge, a se deplasa spre aceasta și a o introduce într-o poartă. Robotul utilizează tehnologii de procesare a imaginilor, navigație autonomă și control al mișcării pentru a îndeplini aceste sarcini. Scopul principal al proiectului este să integreze și să demonstreze funcționalitatea mai multor subsisteme într-un mod coordonat și eficient.

2.1. Detecția de culoare

Primul pas în funcționarea robotului constă în detectarea mingii și a porții. Sistemul folosește o cameră și algoritmi de procesare a imaginilor pentru a identifica culorile specifice asociate cu mingea și poarta (de exemplu, portocaliu pentru minge și albastru sau galben pentru poartă). Acest proces implică:

- Filtrarea culorilor prin tehnici precum segmentarea imaginii în spațiul de culoare HSV.
- Determinarea centrului de masă al obiectelor detectate pentru calcularea poziției lor relative față de robot.

2.2. Deplasarea robotului către mingea de culoare

După identificarea mingii, robotul planifică o traiectorie pentru a ajunge la aceasta. Sistemul integrează datele de la cameră și senzorii de proximitate pentru a:

- Calcula distanța și direcția față de minge.
- Genera comenzi pentru motoare utilizând algoritmi de control precum PID, pentru a asigura o mișcare lină și precisă.

2.3. Împinsul mingii în interiorul porții

Odată ce robotul ajunge la minge, acesta se aliniază față de poartă pentru a împinge

mingea. Acest proces presupune:

- Alinierea poziției robotului astfel încât mingea să fie în direcția porții.
- Ajustarea în timp real a mișcării pe baza poziției actualizate a mingii și a porții, pentru a asigura direcția corectă.

Acest sistem demonstrează capacitatea robotului de a îndeplini sarcini complexe, combinând percepția, navigația și interacțiunea fizică într-un mod autonom.

3. Tehnologii utilizate

Pentru dezvoltarea proiectului Autobot, care simulează funcționalitatea unui robot fotbalist, au fost utilizate următoarele tehnologii și medii de programare:

3.1 Robotul și hardware-ul simulat

- **TurtleBot3**

Robot mobil utilizat în proiecte educaționale și de cercetare.

- În cadrul proiectului, camera frontală este folosită pentru procesarea imaginilor și detectarea mingii și a porții.
- Sistemul de deplasare și navigație permite robotului să identifice și să urmărească mingea, evitând obstacolele din mediul simulat.
- Este echipat cu senzori precum LIDAR și cameră, fiind controlat prin intermediul ROS2.

3.2 Mediul de simulare

- **Gazebo**
  - Simulator 3D care oferă un mediu realist pentru testarea comportamentului robotului.
  - Este utilizat pentru simularea mișcării robotului, a mingii și a porții într-un mediu virtual, replicând condiții similare cu cele din lumea reală.
  - Permite vizualizarea și ajustarea în timp real a parametrilor, cum ar fi traiectoria robotului sau poziția obiectelor.

3.3 Framework-uri și limbaje de programare

- **ROS2 (Robot Operating System 2)**
  - Framework principal pentru coordonarea și controlul robotului.
  - Asigură comunicarea eficientă între componentele robotului, precum camera, motoarele și senzorii, prin intermediul nodurilor ROS și al mesajelor.
  - Este utilizat pentru dezvoltarea funcțiilor de navigație, detecție a mingii și împingerea acesteia către poartă.

- **Python**
  - Limbaj principal utilizat pentru implementarea algoritmilor de procesare a imaginilor și logica de control a robotului.
  - Facilitează integrarea cu ROS2 pentru dezvoltarea nodurilor de detectare și control.

- **OpenCV**
  - Bibliotecă dedicată procesării imaginilor.
  - Utilizată pentru detectarea culorilor, recunoașterea mingii și a porții, precum și pentru urmărirea poziției acestora în mediul virtual.
  - Tehnici precum segmentarea imaginii și detectarea contururilor (ex. filtrul Canny) permit identificarea rapidă și precisă a obiectelor de interes.

Prin utilizarea acestor tehnologii, proiectul Autobot demonstrează capacitatea de a integra componente avansate pentru a realiza sarcini complexe, cum ar fi detectarea, navigarea și interacțiunea autonomă cu obiectele din mediu.

4. Ghidul Programatorului

4.1 Schema bloc generala

Schema bloc generala a aplicatiei poate fi reprezentata astfel:

1. Input: Camera frontala a robotului capteaza fluxul video in timp real.

2. Procesare:

Codul folosete un singur nod, si anume BallFollower. Acest nod este responsabil pentru procesarea imaginilor primite de la o cameră și pentru publicarea comenzilor de viteză pentru a urmări o minge albastră.

3. Output:

Comenzi trimise catre motoarele robotului pentru ajustarea directiei si vitezei.

Structura generala a aplicatiei este organizata in nodul respectiv si componentele robotului:

Numele nodului este definit ca ball_follower_node prin apelul constructorului super().init('ball_follower_node').

Nodul abordează două funcții principale:

Abonarea la topicul /camera/image_raw pentru procesarea imaginilor.

Publicarea pe topicul /cmd_vel pentru controlul mișcării.

5. Ghidul utilizatorului

5. 1 Funcționalitățile aplicației:

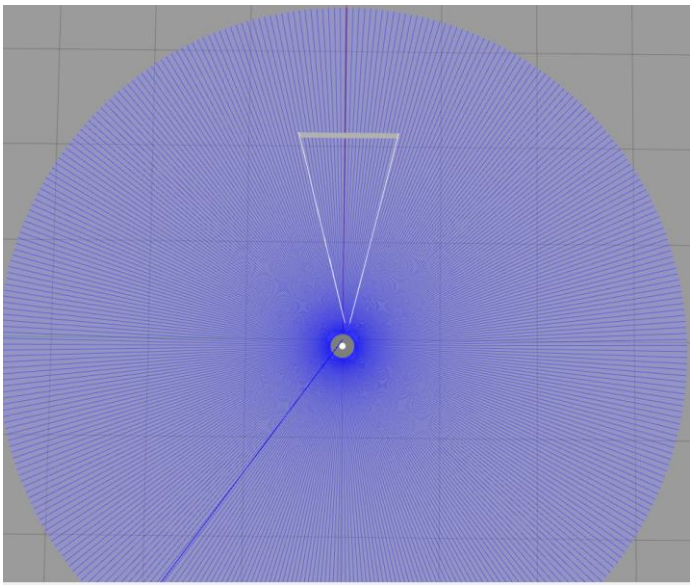
1. **Detectarea mingii**
    - Robotul identifică o minge specifică utilizând camera frontală și algoritmi de procesare a imaginii.
    - Sistemul determină culoarea și poziția mingii, calculând direcția optimă de deplasare.
2. **Navigarea către minge**
    - După detectarea mingii, robotul ajustează direcția și viteza pentru a se deplasa spre aceasta.
    - Traiectoria este calculată în timp real, utilizând algoritmi de navigație bazată pe senzori.
3. **Împingerea mingii în poartă**
    - Robotul se aliniază față de poartă și aplică o forță asupra mingii pentru a o direcționa spre interiorul porții.
    - Sistemul ajustează mișcările pentru a corecta eventualele abateri.

5. 2 Configurare inițială

1. Pornirea robotului în Gazebo (simulare):
Asigurați-vă că mediul ROS2 este configurat corect.
Deschideți un terminal și lansați mediul Gazebo:

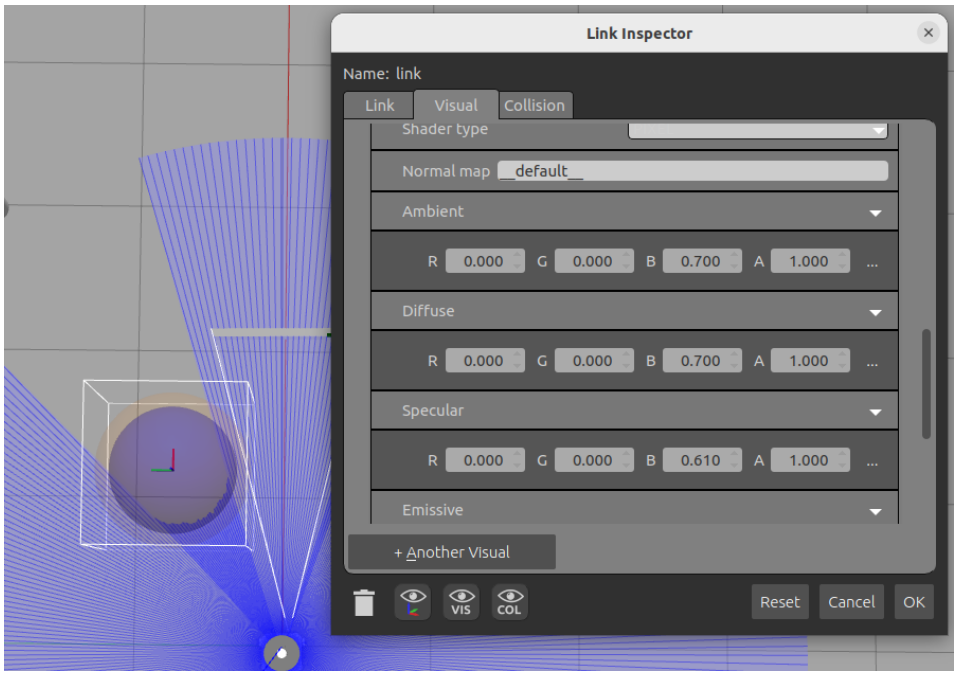
```
vboxuser@ubuntu22:~$ ros2 launch turtlebot3_gazebo empty_world.launch.py
```

Mediul virtual va afișa o lume goala unde putem regasi robotul in mijlocul acesteia.

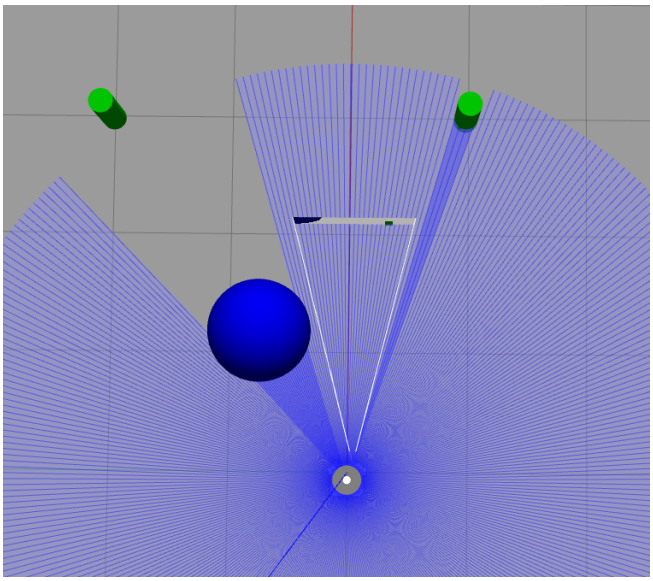


Amplasam mingea, care va fi reprezentata de o sfera de culoare albastra, si doi cilindri de culoare verde, care reprezinta barile portii.

Acestea se selecteaza din tabul din imagine si se amplaseaza in lumea creata, dupa care se editeaza in functie de cerintele necesare. In cazul de fata, mingea o vom face albastra si vom modifica masa si dimensiunea ei pentru a fi mai usor de impins. Pentru a schimba culoarea unui obiect, se da click dreapta pe el, se selecteaza "edit model", apoi "open link inspector", iar dupa de la visual se modifica valorile RGB pt a ajunge la rezultatul dorit.



Acelasi lucru il facem si pentru cele doua bari ale portii.



2. Lansarea aplicației ROS2:
Deschidem un nou terminal si introducem comanda:

```
vboxuser@ubuntu22:~$ ros2 run tb3_tele capture
```

Astfel se va rula codul scris in limbajul python si robotul va executa.

5. 3 Instrucțiuni de utilizare:

1. Detectia mingii dupa culoare ei:
După lansarea aplicației, robotul începe să analizeze fluxul video pentru detectarea mingii in functie de culoarea ei, albastru.
În terminal, veți vedea mesajele:

Acest lucru indica detectia cu succes.

2. Deplasarea și împingerea mingii:
Robotul se deplasează către minge și calculează poziția porții.

```
[INFO] [1737630043.870900692] [ball_follower_node]: Ball follower node started
[INFO] [1737630044.124146752] [ball_follower_node]: Received image frame
[INFO] [1737630044.182113915] [ball_follower_node]: Ball detected at x: 791, rad
ius: 109.5964126586914
[INFO] [1737630044.182606005] [ball_follower_node]: Publishing velocity command
to /cmd_vel: linear.x=0.0, angular.z=0.3
[INFO] [1737630044.417270959] [ball_follower_node]: Received image frame
[INFO] [1737630044.437524459] [ball_follower_node]: Ball detected at x: 791, rad
ius: 109.62446594238281
[INFO] [1737630044.438097044] [ball_follower_node]: Publishing velocity command
to /cmd_vel: linear.x=0.0, angular.z=0.3
```

6. Testare si punerea in functiune

6.1 Modul de testare al aplicatiei:

Testarea aplicației a fost realizată în mediul de simulare Gazebo, utilizând diferite scenarii in care o minge este amplasata aleatoriu pe "lume" iar robotul o detecteaza si o impinge spre poarta.

1. **Testarea detectiei mingii**
   Pentru aceasta functionalitate robotul incepe si face o rotire(twist) pana cand detecteaza mingea in jurul lui, in acel moment va incepe sa se deplaseze spre minge, acest lucru este posibil datorita detectiei de culoare(albastra).

2. **Deplasarea catre minge**
   Dupa detectia mingii(de culoare) robotul va incepe sa se deplaseze spre aceasta pentru a o impinge intr-un final pe directia stabilita.

3. **Testarea fluxului video**
   S-a verificat procesarea corecta a imaginii in timp real cu ajutorul functiei din ros2, camera/image_raw, prin functia "rqt"

6.2 Modul de instalare a aplicatiei

1. **Cerinte preliminare:**

Sistem de operare: Ubuntu 22.04.
ROS2 Humble instalat și configurat.
Simulatorul Gazebo (versiune compatibilă cu ROS2).

2. **Instalarea aplicatiei:**

Compilati si construiti pachetele ROS2:
Rulati comanda "colcon build" in directorul turtlebot3_ws
Dupa lansarea aplicatiei Gazebo si pornirea programului cu comanda:
"ros2 run tb3_tele capture"

7. Concluzii

Proiectul **Fotbalist** reprezintă o implementare practică și educativă în domeniul roboticii autonome, folosind tehnologii moderne precum **ROS2**, **Gazebo** și **OpenCV**.

Acesta a demonstrat capacitatea de a integra multiple componente software și hardware într-un sistem funcțional.

8. Bibliografie

https://docs.ros.org/en/

https://gazebosim.org/

https://opencv.org/

https://docs.python.org/3/

https://docs.ros.org/en/humble/index.htm