INFORMATIKA
TANSZÉK
DEPARTMENT OF INFORMATICS

Széchenyi István Egyetem

Informatikai és Villamosmérnöki Kar

Informatika Tanszék

Féléves feladat

Varga Sámuel

MérnökInformatikus BSc szak

Tartalomjegyzék

Tartalom

Bevezetés

A var_n7k_parkbot egy ROS 2 (Humble) környezetre fejlesztett Python csomag, amely TurtleBot3 mobilrobot szimulált környezetben történő, LIDAR-alapú parkolóhely-felismerését és parkolási logikáját demonstrálja Gazebo szimulátorban, RViz vizualizációval kiegészítve [1] . A rendszer célja egy end-to-end mintastruktúra bemutatása, amely a szimuláció indításától a szenzoradatok feldolgozásán át a vezérlési parancsok kiadásáig lefedi a folyamatot. A dokumentum áttekintést ad a rendszer felépítéséről, fő komponenseiről, a futtatás lépéseiről, a ROS interfészekről, valamint a vizualizációról és a szimulációs világról.

Rendszeráttekintés

A csomag fő komponensei az alábbi funkciókat valósítják meg:

- Parkolási logika: ROS 2 node (parking_logic_node), amely a LIDAR-pontfelhő alapján elérhető, üres parkolóhelyeket detektál és mozgatási parancsokat generál a robot számára [1] .

- Szimuláció: Gazebo világ TurtleBot3 modellel és a szükséges bridge-elt topikokkal.
- Vizualizáció: RViz konfiguráció pontfelhők és detektált parkolóhelyek megjelenítésére.
- Rendszerállapot és transzformációk: robot_state_publisher és tf topikok.
- Parancskiadás és visszajelzések: /model/turtlebot3/cmd_vel sebességparancsok; odometria (/model/turtlebot3/odometry) és LIDAR-pontok (/model/turtlebot3/scan/points).

A README alapján a rendszer topológia a következő fő elemek között valósít meg adatcserét: a parking_logic_node előállítja a vezérlési parancsokat (/cmd_vel), feldolgozza az odometriát és a LIDAR-pontokat, valamint vizuális markereken (/parking_markers) publikálja a detektált parkolóhelyeket. Az RViz a tf és egyéb segédtopikok alapján jeleníti meg az állapotot. A szemléltetéshez lásd a szimulációs világ (Ábra 1), a LIDAR-adatok (Ábra 2) és a klaszterezett detekció (Ábra 3) képeit, melyek a csomagban találhatók.

Projektstruktúra

Az alábbi táblázat a repository főbb elemeit foglalja össze a feladathoz kapcsolódó szempontok szerint.

A repository fő elemei és szerepük (forrás: [1])

Elérési út	**Típus**	**Leírás**
CMakeLists.txt	fájl	Build-konfiguráció (ament/cmake integráció a ROS 2 csomaghoz).
package.xml	fájl	ROS 2 csomagmetaadatok, függőségek deklarációja.
setup.py, setup.cfg	fájl	Python csomag telepítési és konfigurációs fájlok.
var_n7k_parkbot/	könyvtár	A Python forráskód és node-ok helye (pl. parking_logic_node entrypoint).
launch/	könyvtár	Indítási (launch) fájlok, pl. Gazebo + RViz + bridge-ek indítása.
rviz_config/	könyvtár	RViz konfigurációk a vizualizációhoz.
robot_description/	könyvtár	Robotleírások (URDF/Xacro) és kapcsolódó erőforrások.
world/	könyvtár	Gazebo világok a parkolási jelenetekhez.
teleop_test/	könyvtár	Teleoperációs tesztekhez kapcsolódó anyagok.

Elérési út	Típus	Leírás
img/	könyvtár	A dokumentumban hivatkozott képernyőképek (Ábrák).
README.md	fájl	Rövid projektleírás, gyors indítás és topológia-diagram.
LICENSE	fájl	Licencfeltételek.
.gitignore	fájl	Verziókezelési kizárások.
test/	könyvtár	Tesztek.

Telepítés és futtatás

A csomag a README szerint ROS 2 Humble környezetben készült, colcon build eszközlánccal [1] , [7]

Előfeltételek

- ROS 2 Humble (ajánlott a hivatalos disztribúció telepítése) [2]
- Gazebo szimulátor (a ROS 2-vel kompatibilis csomagokkal) [3]
- TurtleBot3 csomagok a szimulációhoz [4]
- RViz 2 a vizualizációhoz [5]
- colcon a buildeléshez [7]
- Python 3 (a ROS 2 Humble kompatibilis verzió)

Lépések (összefoglaló)

- Klónozás: a csomagot a ROS 2 workspace src mappájába klónozzuk.
- Build: colcon build --packages-select var_n7k_parkbot --symlink-install
- Forrásolás: source ~/ros2_ws/install/setup.bash
- Indítás: ros2 launch var_n7k_parkbot gazebo_with_robot.launch.py
- Parkolási logika node: külön terminálból ros2 run var_n7k_parkbot parking_logic_node

Fő futtatási parancsok (forrás: [1])

Lépés	Parancs
Klónozás	cd ~/ros2_ws/src && git clone https://github.com/samuvarga/var_n7k_parkbot

Lépés	Parancs
Build	cd ~/ros2_ws && colcon build --packages-select var_n7k_parkbot --symlink-install
Forrásolás	source ~/ros2_ws/install/setup.bash
Indítás (Gazebo + RViz + bridge-ek)	ros2 launch var_n7k_parkbot gazebo_with_robot.launch.py
Parkolási logika indítása	ros2 run var_n7k_parkbot parking_logic_node

Megjegyzések:

- A launch fájl elindítja a Gazebo világot, beállítja a bridge-elt topikokat és megnyitja az RViz-t.
- A parkolási logika külön terminálban indítandó.

ROS interfészek (topikok és komponensek)

A README Mermaid-diagramja alapján a következő fő topikok és komponensek vesznek részt a rendszerben. A topikok üzenettípusai a README-ben nem részletezettek; a táblázat a funkció szerinti szerepkört foglalja össze.

Fő ROS topikok és szerepük (forrás: [1])

Név	Szerep	Megjegyzés
/model/turtlebot3/cmd_vel	Vezérlési parancsok	A parking_logic_node publikálja; a bridge kezeli a Gazebo modell felé.
/model/turtlebot3/odometry	Visszajelzés	Odometria; a parking_logic_node felhasználja.
/model/turtlebot3/scan/points	Szenzoradat	LIDAR pontfelhő; a parking_logic_node felhasználja.
/model/turtlebot3/parking_markers	Vizualizáció	Detektált parkolóhelyek marker-üzenetei RViz-hez.

Név	Szerep	Megjegyzés
/tf	Transzformációk	robot_state_publisher és RViz használja.
/robot_description	Robotleírás	Robot állapota és kinematikai leírás.
/joint_states	Ízületállapotok	robot_state_publisher bemenet.
/clock	Szimulációs idő	Gazebo órája a ROS 2-nek.
/parameter_events	Paraméteresemények	Komponensek paraméterezési eseményei.
/initialpose, /goal_pose, /clicked_point	Interaktív RViz inputok	RViz-ből érkező interakciós topikok.
/rosout	Naplózás	Komponensek naplóüzenetei.

Parkolási logika – áttekintés

A parkolási logika a LIDAR-pontfelhőt dolgozza fel, klaszterezéssel meghatározza a potenciális akadályok és parkolóhelyek elhelyezkedését, majd vezérlési parancsokat ad ki a robotnak a /cmd_vel topikon keresztül a parkolási manőver végrehajtására. A működés fő lépései:

- Szenzoradatok fogadása: LIDAR pontfelhő (/scan/points) és odometria (/odometry).
- Előfeldolgozás: pontfelhő szűrése és szegmentálása (pl. föld-sík eltávolítása, zajszűrés).
- Klaszterezés: akadályok és potenciális parkolóhelyek elkülönítése (Ábra 3).
- Parkolóhely-értékelés: geometriák és biztonsági távolságok ellenőrzése.
- Mozgástervezés és vezérlés: sebességparancsok kiadása (/cmd_vel) a kívánt pályára való beálláshoz.
- Vizualizáció: detektált helyek marker-üzenetekkel történő publikálása RViz felé (/parking_markers).

Megjegyzés: a részletes algoritmikai megvalósítás a forráskódban található (var_n7k_parkbot mappa), a README magas szintű áttekintést nyújt.

Vizualizáció RViz-ben

A rendszer RViz-ben jelzi a LIDAR pontfelhőt és a detektált parkolóhelyeket (markerek) [1]. A szemléltetéshez lásd az Ábra 2 és Ábra 3 képeket; ezek alapján ellenőrizhető a szenzoradatok minősége és a detekció kimenete.

- LIDAR pontfelhő megjelenítése: színkódolt pontok a környezet akadályainak reprezentációjával (Ábra 2).
- Detektált klaszterek és parkolóhelyek: marker-objektumok a találatok szemléltetésére (Ábra 3).

Gazebo szimulációs világ

A csomag tartalmaz parkolási jelenethez készült Gazebo világot, ahol a TurtleBot3 modell manőverezik. A megfelelő bridge-elt topikok biztosítják a ROS 2 és Gazebo közti kommunikációt (cmd_vel, odometry, LIDAR-pontok), a szimulált környezet pedig lehetővé teszi a reprodukálható kísérleteket és demókat. A világ vizuális képe a csomag képei között elérhető (Ábra 1).

Teleoperációs tesztelés

A teleop_test könyvtár a kézi vezérléshez és gyors funkcionális tesztekhez biztosíthat anyagokat. Teleoperációval ellenőrizhető:

- a mozgásparancsok hatása,
- az odometria konzisztenciája,
- a vizualizációs pipeline helyes működése (pontfelhő és markerek megjelenése).

Naplózás és hibakeresés

A /rosout topikra publikált naplóüzenetekből követhető a komponensek állapota. Hibakereséshez javasolt:

- rviz2 kijelzők ellenőrzése (tf, LaserScan/PointCloud2, Marker/MarkerArray),
- a bridge-elt topikok jelenlétének vizsgálata (ros2 topic list/echo),
- a node paraméterek és események figyelése (/parameter_events),
- a szimulációs óra működésének ellenőrzése (/clock).

Korlátok és jövőbeli bővítések

Lehetséges bővítési irányok:

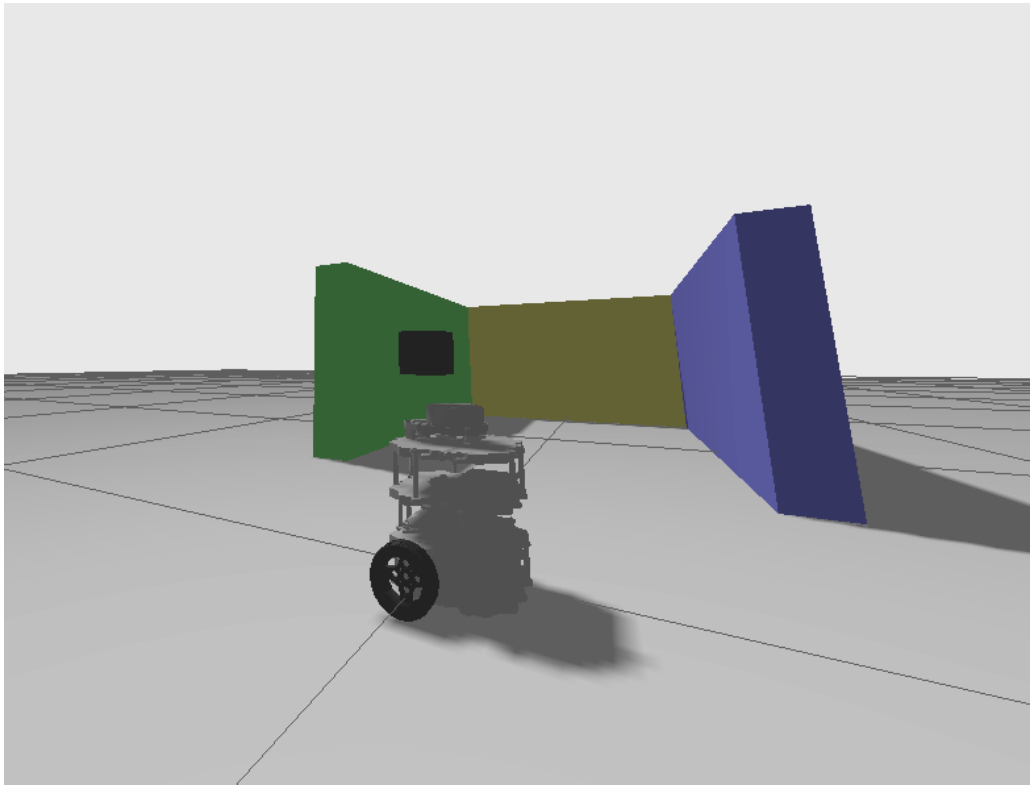
- Robusztusabb klaszterezés és zajkezelés különböző környezeti feltételekhez.
- Parkolóhely-értékelés pontosítása (geometriai és kinematikai korlátok explicit kezelése).
- Magasabb szintű tervezők integrációja (útvonalterv a parkolóhelyhez) és ütközéselkerülés.
- Paraméterezhetőség és konfigurációs profilok kiterjesztése.
- Valódi robotra történő átültetés és szenzorfúzió (például vizuális megerősítés).

Licenc és jogi nyilatkozat

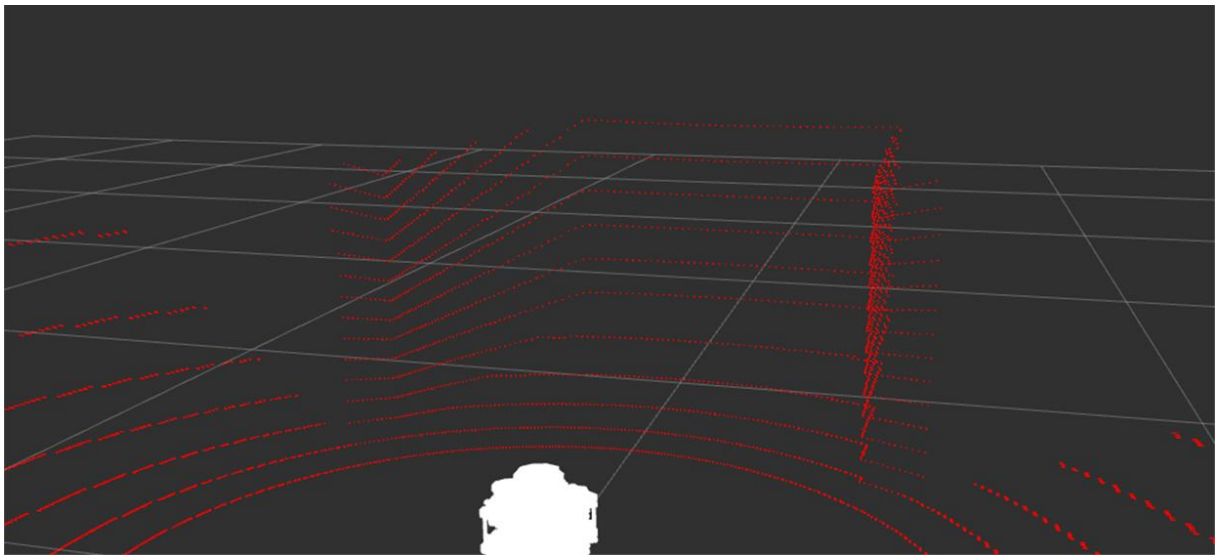
A projekt licencfeltételei a repoository LICENSE fájljában találhatók. A külső komponensek (ROS 2, Gazebo, TurtleBot3, RViz stb.) saját licencek alatt érhetők el [2]-[5] . A felhasználás során vegye figyelembe ezek feltételeit.

Ábrák

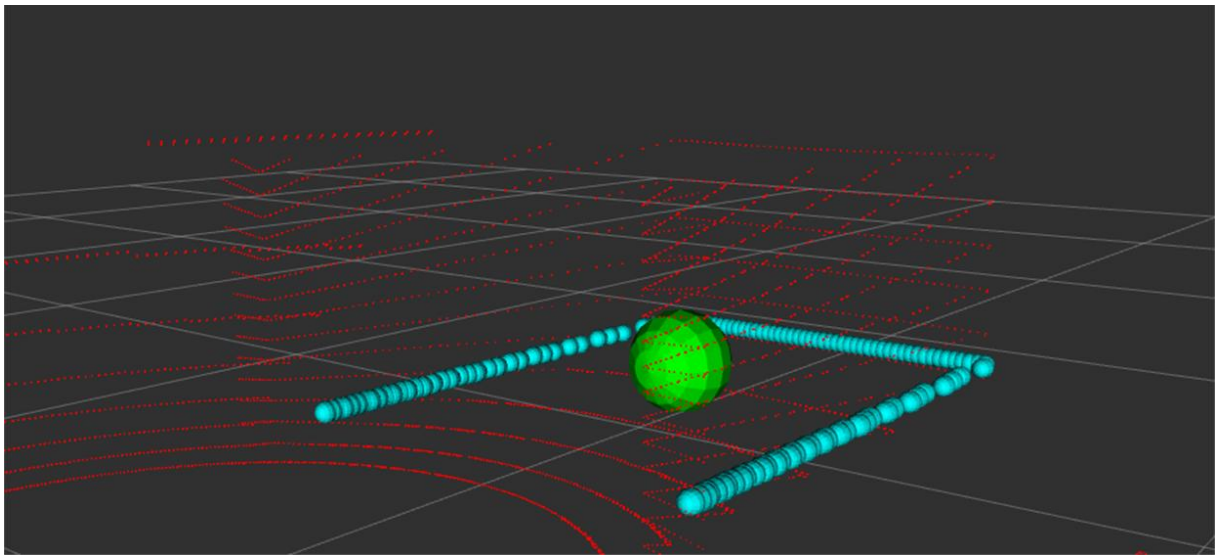
Ábra 1 Szimulációs világ képe (forrás: [1])



Ábra 2 LIDAR pontfelhő RViz-ben (forrás: [1])



Ábra 3 Detektált klaszterek és parkolóhelyek (forrás: [1])



Irodalomjegyzék

[1] S. Varga, var_n7k_parkbot – ROS 2 Humble Python csomag TurtleBot3-hoz. Elérhető: https://github.com/samuvarga/var_n7k_parkbot (lekérés dátuma: 2025.11.28)

[2] Open Robotics, ROS 2 Humble dokumentáció. Elérhető: https://docs.ros.org/en/humble/

[3] Gazebo Simulator projekt. Elérhető: https://gazebosim.org/ (ROS integráció: https://gazebosim.org/docs)

[4] Robotis, TurtleBot3 dokumentáció. Elérhető: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

[5] RViz 2 – ROS 2 vizualizációs eszköz. Elérhető: https://docs.ros.org/en/humble/Tutorials/Intermediate/RViz2.html

[6] PCL/PointCloud2 és LIDAR pontfelhő feldolgozás ROS 2-ben. Áttekintő források: https://pointclouds.org/; ROS msg: sensor_msgs/PointCloud2

[7] colcon – Ament/ROS 2 build eszköz. Elérhető: https://colcon.readthedocs.io/