**Bevezetés és előfeltételek**

Ez a jegyzet az E-Puck robotra szerelt LIDAR (Light Detection and Ranging) távolságmérő szenzor használatát mutatja be a Webots szimulációs környezetben. A jegyzet feltételezi, hogy a szenzor már hozzá lett adva a robothoz a szimulátorban, és a vezérlő (controller) szkript inicializációs részében engedélyezve lett az alábbi Python kódsorokkal:

# A LIDAR eszköz inicializálása

lidar = robot.getDevice('LDS-01')

lidar.enable(timestep)

lidar.enablePointCloud()

A jegyzet célja, hogy segítsen a szimulációs környezet helyes beállításában és a távolságmérő eszköz működésének megértésében.

**A LIDAR szenzor működése és adatfeldolgozása**

**1. Kérdés: Az adatok lekérdezése és mennyisége**

A szenzor által mért távolságadatokat, az úgynevezett "range image"-et, az alábbi paranccsal lehet lekérdezni:

ranges = lidar.getRangeImage()

Hány adatpontot kapunk a szenzortól egyetlen mérés során?

**Válasz:** 360

**2. Kérdés: A mérési pontok orientációja**

Merre mutat a LIDAR tömb első sugara (a ranges[0] elem)?

* A robot eleje felé, egyenesen az X-tengely mentén.
* A robot hátulja felé, egyenesen a negatív X-tengely mentén.
* A robot jobb oldala felé, egyenesen a negatív Y-tengely mentén.
* A robot bel oldala felé, egyenesen az Y-tengely mentén.

**3. Kérdés: A szenzorsugarak eloszlása**

**Kérdés:** Hogyan oszlanak el a sugarak a robot körül (a 0. ponthoz képest)?

* Az óramutató járásával megegyező irányban (Clockwise).
* Az óramutató járásával ellentétes irányban (Counter-clockwise).

**4. Kérdés: Szögek hozzárendelése a mérésekhez**

Ahhoz, hogy az akadályok helyzetét a robot koordináta-rendszerében kiszámíthassuk, minden távolságadathoz hozzá kell rendelnünk egy szöget. Erre a NumPy linspace() funkciója kiválóan alkalmas.

**Kérdés:** Melyik adatszerkezet segítségével tudjuk a ranges lista minden eleméhez a megfelelő szöget hozzárendelni (radiánban), feltételezve, hogy a 0 fok (0 radián) a robot pozitív X-tengelyével esik egybe?

Javasolt kipróbálni a kódot válaszadás előtt. Kirajzoláshoz a következő kódrészletet lehet használni (polárkoordinátás megjelenítés). A szimuláció a plot miatt meg fog állni, az ablak bezárásával lép előre.

from matplotlib import pyplot as plt

fig, ax = plt.subplots(subplot\_kw={'projection': 'polar'})

ax.plot(angles,ranges,'.')

plt.show()

**Válasz:**

* angles = np.linspace(-3.1415,3.1415)
* angles = np.linspace(-3.1415,3.1415,360)
* angles = np.linspace(3.1415,-3.1415)
* angles = np.linspace(3.1415,-3.1415,360)

**5. Kérdés: Pontok koordinátáinak kiszámítása**

A távolságadatok (ranges) és a hozzájuk rendelt szögek (angles) segítségével kiszámíthatjuk az egyes mérési pontok (x, y) koordinátáit a robot saját, Descartes-féle koordináta-rendszerében.

**Kérdés:** Melyik kifejezés alkalmas egy i indexű mérési pont (xᵢ, yᵢ) koordinátáinak kiszámítására?

# A teszteléshez és plotoláshoz az alábbi kódrészletet lehet felhasználni:

if(np.isfinite(ranges[i])):

...

# Azokat a távolságértékeket használjuk, amelyek végesek. Amennyiben több értéket kell megjeleníteni rendezzük listába:

x,y =[], []

...

x.append(x\_i)

y.append(y\_i)

# Kirajzoláshoz:

plt.plot(x,y,'.')

**Válasz:**

* x\_i = np.cos(angles[i]) \* ranges[i]

y\_i = np.sin(angles[i]) \* ranges[i]

* x\_i=np.sin(angles[i])\*ranges[i]

y\_i=np.cos(angles[i])\*ranges[i]

* x\_i=(np.cos(angles[i])-sin(angles[i]))\*ranges[i]

y\_i=(np.sin(angles[i])+np.cos(angles[i]))\*ranges[i]

* x\_i=np.cos(angles[i])\*ranges[i]\*np.cos(theta)

y\_i=np.sin(angles[i])\*ranges[i]\*np.sin(theta)

**Koordináta-transzformáció és szenzorparaméterek**

Ez a rész feltételezi a lézerszenzor adat-koord.transzformálásának ismeretét.

**6. Kérdés: A robot világkoordinátáinak frissítése**

**Kérdés:** Fordítsuk el a robotot a szimuláció elején 180 fokkal, hogy lefelé haladjon (lehet, hogy kissé el kell tolni a startvonaltól). Mit kell tennünk, hogy a lézerszenzor koordinátái továbbra is helyesen legyenek kiszámítva a világkoordináta-rendszerben?

* Vissza kell állítani a theta értékét nullára a kód elején.
* Frissíteni kell az xw, yw és theta alapértelmezett értékeit a kódban.
* Minden trigonometrikus egyenletben hozzá kell adni π-t a theta értékéhez.

**7. Kérdés: A látómező (Field of View) módosítása**

**Kérdés:** A lézeres távolságmérő paraméterei (pl. a látómező) megváltoztathatók a Webotsban. Hogyan kell módosítani a kódot, ha a látómezőt (field of view) 6.28 radiánról (~360°) 1.5758 radiánra (~90°) csökkentjük?

* Frissíteni kell a szögek tömbjét: angles=np.linspace(1.5758/2, -1.5758/2, 90)
* Semmit nem kell változtatni, mivel a szögeket minden méréssel együtt kapjuk meg.
* Frissíteni kell a szögek tömbjét: angles=np.linspace(1.5758/2, -1.5758/2, 360)

**8. Kérdés: A szenzor frissítési rátájának beállítása**

**Kérdés:** A valódi Robotis LDS-01 szenzor frissítési frekvenciája 5 Hz. Milyen x értéket kellene megadni a lidar.enable(x) parancs paramétereként, hogy ezt szimuláljuk? (A szimuláció alap idő-lépése (timestep) adott.)

* 5
* 32
* 200
* 5000

**9. Kérdés: A frissítési ráta és a szimulációs időlépés viszonya**

**Kérdés:** Mi történik, ha a lézerszenzor frissítési rátáját a szimuláció 32 ms-os alap idő-lépésénél gyorsabbra állítjuk (pl. kisebb értéket, mondjuk 16-ot adunk meg az enable() funkciónak)?

* A szimuláció első néhány lépésében hibás adatokat kapunk, majd a szenzor adatai egyszerűen késni fognak.
* A szenzor adatai eltorzulnak vagy eldeformálódnak, és a hatás annál erősebb, minél nagyobb az eltérés a szimulációs idő és a frissítési ráta között.
* Minden olyan időlépésben hibás adatokat kapunk, ahol a szimulációs időlépés és a szenzor frissítési rátája nincs szinkronban.

**Koordináta-transzformáció világkoordináta-rendszerbe**

Az előző részben kiszámítottuk a lézerszenzor által észlelt akadályok pozícióit a robot saját koordináta-rendszerében. Navigációs és térképezési feladatokhoz azonban ezeket a pontokat egy globális, fix világkoordináta-rendszerben kell ismernünk. Ezt a transzformációt homogén mátrixok segítségével végezhetjük el hatékonyan.

A homogén transzformáció lényege, hogy egyetlen mátrixszorzással képesek vagyunk végrehajtani a forgatást (rotáció) és az eltolást (transzláció) is. Ehhez a 2D-s pontjainkat (x, y) egy harmadik, úgynevezett homogén koordinátával egészítjük ki, így [x, y, 1] formájú vektorokat kapunk.

A pontok mátrixa (**X\_i**): Az összes szenzormérésből származó, robot-koordinátarendszerbeli pontot egyetlen mátrixba rendezzük. Ennek a mátrixnak 3 sora lesz: az első az x, a második az y koordinátákat, a harmadik pedig a homogén 1-eseket tartalmazza.

X\_i = np.array([ranges \* np.cos(angles),

ranges \* np.sin(angles),

np.ones(len(angles))])

A transzformációs mátrix (**w\_T\_r**): Ez egy 3x3-as mátrix, amely leírja a robot pozícióját (xw, yw) és orientációját (theta) a világkoordináta-rendszerhez képest.

w\_T\_r = np.array([[np.cos(theta), -np.sin(theta), xw],

[np.sin(theta), np.cos(theta), yw],

[0, 0, 1]])

* A bal felső 2x2-es rész a forgatómátrix, amely a theta szög alapján elforgatja a pontokat.
* A jobb felső 2x1-es oszlop (xw, yw) az eltolási vektor, amely a robot pozíciójával tolja el a pontokat.
* Az alsó [0, 0, 1] sor a homogén számításokhoz szükséges.  
  A w\_T\_r jelölés azt fejezi ki, hogy a transzformáció a robot (r) rendszerből a világ (w) rendszerébe képez le.

A transzformáció elvégzése: A világkoordinátákat (D) egyetlen mátrixszorzással kapjuk meg:

D = w\_T\_r @ X\_i

Az eredmény, D, amelynek oszlopai az akadályok pozícióit tartalmazzák a világkoordináta-rendszerben.

**Kvízkérdések a transzformációhoz**

**10. Kérdés: A transzformációs mátrix azonosítása**

**Kérdés:**Melyik adatszerkezet a "homogén transzformációs mátrix" a fenti kódban?

* [[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]]
* w\_T\_r
* X\_i

**11. Kérdés: Matematikai jelölés**

**Kérdés**: Hogyan írható fel a D = w\_T\_r @ X\_i művelet matematikai jelöléssel?

* D = ʳ\_w\_T \* Xᵢ
* D = ʷ\_r\_T Xᵢ
* D = wᵀ Tᵣ Xᵢ

**12. Kérdés: A transzformációs mátrix felépítése**

**Kérdés**: Mi a w\_T\_r helyes matematikai felírása?

* [cos(θ) sin(θ) xw]  
  [-sin(θ) cos(θ) yw]  
  [0 0 1 ]
* [cos(θ) -sin(θ) xw]  
  [sin(θ) cos(θ) yw]  
  [0 0 1 ]
* [cos(θ) -sin(θ) 0 xw]  
  [sin(θ) cos(θ) 0 yw]  
  [0 0 1 0]  
  [0 0 0 1]

**13. Kérdés: A pontmátrix dimenziója**

**Kérdés**: Mekkora a X\_i mátrix dimenziója?

* 3x360
* 3x1
* 360x3

**14. Kérdés: A pontmátrix tartalma**

**Kérdés**: Mit tartalmaz az X\_i tömb?

* A tömb azoknak a pozícióknak a listáját tartalmazza, ahol az egyes szenzorsugarak akadályba ütköztek, a robot koordináta-rendszerében kifejezve.
* A tömb azoknak a pozícióknak a listáját tartalmazza, ahol az egyes szenzorsugarak akadályba ütköztek, a világkoordináta-rendszerében kifejezve.
* A tömb a távolságokat és szögeket tartalmazza minden méréshez, valamint egy harmadik sort kiegészítésként.