aiMotive próbafeladat

0. A program működése

A programnak parancssori argumentumkén kell átadni a egy csv fájl vagy annak a mappának az elérési útját, ami a csv fájlokat tartalmazza.

Ezekután egy 4 menüpontból választhatunk, amit a menüpont sorszámát beírva adhatunk meg. Ezek a következőek:

0 --> A program leállítása.

1 --> Teljes statisztika futtatása.
Ha csak egy csv fájlt kapott a program, akkor csak annak az egy frame-nek a statisztikáit láthatjuk, ha egy mappát adtunk meg mint elérési cél, akkor a mappában található összes frame-ről kapunk statisztikálat.
Ezek a következőket tartalmazzák:
Általános adatok:
- A fájl elérési útja
- Pontok, ahol x, y és z koordináták a maximumot, illetve a minimumot felveszik
- pontok száma az őket detektált lézer ID-k szerint, valamint a hibásan detektált pontok száma

A fal érzékelésével kapcsolatok adatok:
- A fal pontjaira illesztett sík egyenlete
- Az átlagos eltérés a fal pontjai és a falra illesztett sík között

A henger érzékelésével kapcsolatos adatok:
- A henger magassága
- A henger szélessége,
- A henger magasságának és szélességének az aránya.

2 --> Csak a fal érzékelésévek kapcsolatos adatok megjelenítése.

3 --> Csak a henger érzékelésével kapcsolatos adatok megjelenítése.

Az összes csv fájl beolvasása és feldolgozása egész időigényes ezért a teljes statisztika kimenetét a projekten belüli results mappába elmentettem egy szöveges fájlba. Az egyes Frame-hez tartozó statisztikákat külön szöveges fájlokba is kimentettem. Ezek az eredeti csv fájlok neveit kapták.

1. Adatok beolvasása és feldolgozása

Az adatok feldolgozásáért és beolvasásáért felelős osztályokm illetve függvények a sensorParser.py fájlban vannak.

Még a main.py fájlon belül a main függvényben ellenőrzőm, hogy a program indításkor megkapta-e parancssori algoritmusként a kívánt fájl/mappa elérési útvonalát. Amennyiben nem ezt jelzi a felhasználó felé és leáll a program. Ha igen ellenőzi, hogy az elérési út egy mappára vagy egy fájlra mutat-e. Amennyiben csak egy fájlra, akkor egy egyelemű listát csinálok belőle. Ammenyiben egy mappára mutat az útvonal, akkor átadom az elérési útvonalat a **file_list_from_dir** nevű függvények.

A sensorParser.py fájlban található **file_list_from_dir** függvény egy mappa elérési útvonalát várja paraméterként, illetve opcionálisan megadható, hogy milyen kiterjesztésű fájlokat keresünk. Alapesetben .csv kiterjesztésű fájlokat keres. A függvény végigmegy a mappában található összes csv fájlon és az elérési útvonalaikat kigyűjti egy listába. Ez a lista a függvény visszatérési értéke.

A fájl eléréséket tartalmazó listát átadom a **frame_list** nevű függvénynek. Ez betülti a listában található összes fájlt. Egy-egy fájl nyers szövegét átadja a **point_list** nevű függvénynek, ami a vesszők mentén feldarabolja a szöveg minden sorát. Ezt a feldarabolt szöveget tartalmazó listát, valamint a fájl elérését paraméterül adva létrehoz minden egyes példányt a Frame osztályból. A frame_list függvény visszatérési értéke egy lista, ami az összes Frame objektumot tartalmazza.

A Frame osztály tárolja egy Frame adatait, mit a Frame-ben található pontok listája, valamint annak a cvs fájlnak az elérési utját, amiből a Frame objektum készült.

A Point osztály egy Frame egy pontját tárplja. Ezek a például az intenzitás, a lézer ID, távolság, x, y, z koordináták stb.

2. Hibák az adatokban

Komolyabb hibákat nem vettem észre a ponthalmazokban.
Minden Frame-ban voltak hibás pont adatok. Ez abban nyívánult meg, hogy egy pont minden értéke ki volt nullázva. Én ezeket mérési hibának vélte. Egyik frame esetében sem volt a hibák aránya nagy, ezekről statisztika is van különk lézet ID-k szerint bontva.
Például a 100. frame esetén ez a következő képpen néz ki:

Laser ID: 0	Number of points: 15750	Number of fails: 186
Laser ID: 1	Number of points: 15750	Number of fails: 105
Laser ID: 2	Number of points: 15750	Number of fails: 103
Laser ID: 3	Number of points: 15750	Number of fails: 86
Laser ID: 4	Number of points: 15750	Number of fails: 0

Másik észrevétel, hogy bizonyos ponthalmazok mérete kisseb. Ezek a 0-ás, a 93-as és a 97-es Framek.
Ezekben az esetekben a szoba felső részének a pontja hiányoznak, így például a henger valódi magassága ezeknél a Frame-nél nem lehetet meghatározni.

Nem hiba, inkább csak egy további észrevétel, hogy minden ponthalmaz tartalmaz pontokat a szobán kívülről is. Valószínűleg egy ablakon vagy egy ajtón keresztül a szomszédos szobába is bemért a LiDAR.

3. A fal érzékelése

A fal illetve a henger érzékeléséhez szükséges függvények illetve osztályok a statistics.py fájlban találhatóak.

A WallDetection osztály tárolja a fal érzékélésével kapcsolatok adatokat. Paraméterként a pontoknak azt a halmazát várja, amik a falhoz tartoznak. Ezeket a pontokat egyszerű konstansokkal határoztam meg hogy ebben az esetben mely területre esnek és így válogattam ki őket.

A pontok halmazára állít egy közelítő síkot. Ezt a plane_from_points függvény végzi, ami egy pontokat tartalmazó listát vár. A visszatérési értéke egy lista, ami a sík A, B és D paramétereit tartalmazza (Z = A x + B Y + D).

Ezekután a pontok és a sík közötti átlagos távolságot számítja ki a program. Ezt az avg_dev_point_plane függvény számítja ki. Paraméterként a sík A, B, D paramétereit tartalmazó listát, illetve a pontok listáját várja. Kiszámítja minden pontra a síktól való távolságot és átlagolja. Visszatérési érték egy float, ami az átlagos eltérést tartalmazza

A 100. Frame esetében a sík egyenlete:

z = 14.271314303522864x + 0.4309804603256507y + -84.4342849298883

valamint a pontok és a sík átlagos távolsága

0.032826682971270 89

Az átlagos távolságból láthatjuk, hogy a szenzor egész jól érzékelte a falat, csupán pár milliméteres az eltérés a síktól.

4. A henger érzékelése

A CylinderDetection tárolja a henger érzékelésével kapcsolatok adatokat. Paraméterekként a pontoknak azt a két halmazát várja, amik a hengerhez illetve a falhoz tartoznak. A hengerhez tartozó pontokat is ugyan úgy mint a fal pontjait egyszerűen meghatározom hogy melyik területen vannak.

A get_height függvény segítségével kiszámolom a henger magasságát. Paraméterként megkapja a hengerhez tartozó pontok listáját. A pontok közül megkeresem a legnagyobb, illetve a legkisebb z értékű pontot. A két pont z koordinátájának az értékének a különbsége megadja a henger magasságát. A függvény visszatérési értéke a henger magassága.

A henger átmérőjének a kiszámítása kicsit nehezebb. Ezt a get_width nevű osztály végzi. A függvény paraméterként megkapja a hengerhez tartozó pontok listáját, valamint a falhoz tartozókat is. A henger a falon árnyékot hagy. Ennek az árnyéknak meghatározom a kér szélét. A két széle és az origó között állítok egy-egy egyenest. Az egyenesek 2D-ben vannak, a z koordinátát nem kell figyelembe venni. Az egyik egyenes lévő legközelebbi pontot megkeresem a henger pontjait tartalmazó listából. Ez a pont és a másik egyenes közötti távolság adja a henger átmérőjét. A visszatérési érték ez az átmérő.

A 100-as Frame esetén a henger magassága:

1.053003119478437

A henger átmérője:

0.10978103167526145

A feladat leírásában szerepelt, hogy a henger pontosan 1 méter magas és 10 cm széles. Azaz a magasság/szélesség aránya:

0.1

Ehhez képest a LiDAR adatokból kiszámolt magasság/ szélesség aránya:

0.10425518181716031

Jól látszok hogy az elvárt, valamint a kapott érték közötti eltérés csupán 4% azaz a LiDAR egész pontosan érzékeli a hengert is.