Dokumentácia – Mapovanie

Úloha

Úlohou mapovania je vytvoriť reprezentáciu prostredia na základe údajov zo snímačov robota. Inak povedané, mapovanie je fúzia (spájanie) informácií o polohe robota s informáciami o prekážkach získaných z laserového diaľkomera.

Algoritmus

Algoritmus je proces mapovania robota, kde sa robot pomocou LIDARa pohybuje v prostredí a nepretržite mapuje oblasť okolo neho. Tento proces umožňuje plánovanie cesty, vyhýbanie sa prekážkam a postupom SLAM (Simultaneous Localization and Mapping):

(Pohyb robota ja spracovaný podobne ako v úlohe 1)

- Nastavenie orientácie: Uhol robota je nastavený tak, aby sa zabezpečilo, že je v kladnej radiánovej miere (rad), čo je dôležité pre nasledujúce výpočty.

- Spracovanie údajov: Za určitých podmienok súvisiacich s pohybom robota a mierou odchýlky (odchylka_pol) kód spracováva aktuálne dáta skenovania LIDARa. Pre každý bod skenovania, ak je v prijateľnom rozsahu (okrem špecifického intervalu), sa jeho súradnice (vzdialenosť a uhol od robota) prevedú na karteziánske súradnice (Xgi, Ygi) vzhľadom na polohu a orientáciu robota. Tieto súradnice zodpovedajú bunke v mriežke obsadenosti, ktorá je označená ako obsadená (occ_grid[Xgi][Ygi] = 1).

```
379     if(mojRobot.translation != 0 && odchylka_pol > 0.5 && (odchylkaCelkova*0.97 < odchylka_pol)){
380
381         for(int k=0;k<copyOfLaserData.numberOfScans;k++){
382             scanDist = copyOfLaserData.Data[k].scanDistance / 1000;
383
384             if(((copyOfLaserData.Data[k].scanDistance/1000) < 0.23) && (copyOfLaserData.Data[k].scanDistance/1000) != 0) i++;
385
386
387             if((scanDist > 0.13) && (scanDist < 3.0) && !((scanDist < 0.7) && (scanDist > 0.62))){
388                 lidarAngle = copyOfLaserData.Data[k].scanAngle * (pi1/180.0);
389                 calAngle = rads - lidarAngle;
390                 if(calAngle > 6.283185) calAngle -= 6.283185;
391                 ///printf("MERAM");
392                 Xgi = gSize/3 + (int)((mojRobot.x + scanDist*cos(calAngle))/0.05);
393                 Ygi = gSize/3 + (int)((mojRobot.y + scanDist*sin(calAngle))/0.05);
394                 occ_grid[Xgi][Ygi] = 1;
395             }
396         }
397     }
```

Obr. 1

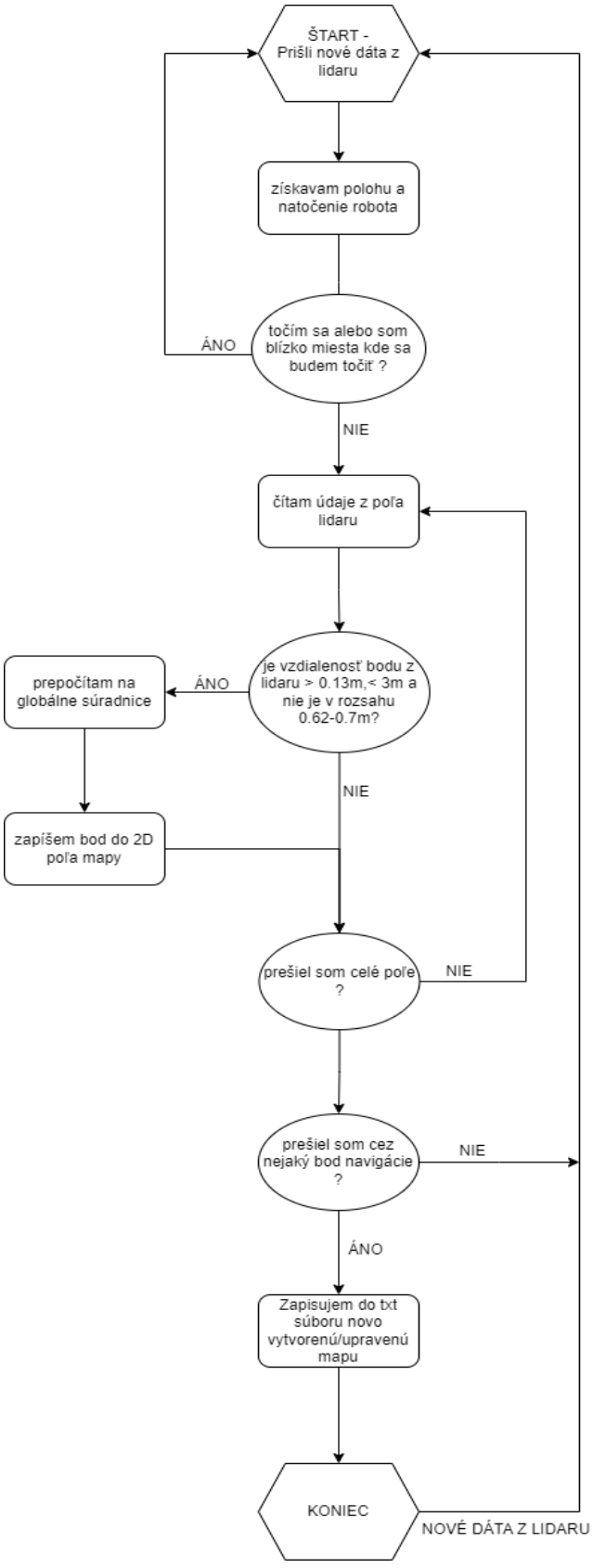
- Aktualizácia mriežky: Keď sa zmení počet bodov, ktoré robot sleduje, zaznamená aktuálnu mriežku obsadenosti do dvoch súborov: jeden v

riadkovom formáte (occGrid.txt) a jeden vo formáte čitateľnom pre ľudí (mapa.txt). Môže sa použiť na ladenie alebo vizualizáciu.

```
405         if(prevNumOfPoints != mojRobot.numOfPoints){
406             odchylkaCelkova = odchylka_pol;
407             ofstream mapa("C:/Users/lukac/Desktop/RMR/RMR2023/uloha3/mapa.txt");
408             ofstream occGrid("C:/Users/lukac/Desktop/RMR/RMR2023/uloha3/occGrid.txt");
409             printf("Zapisujem do mapy");
410             for (int i = 0; i < gSize; i++) {
411                 for (int j = 0; j < gSize; j++) {
412                     occGrid << occ_grid[j][i];
413                     if(occ_grid[j][i] == 0){
414                         mapa << " ";
415                     }else mapa << "x";
416                 }
417                 occGrid << endl;
418                 mapa << endl;
419             }
420             occGrid.close();
421             mapa.close();
422         }
```

Obr. 2

Diagram Algoritmu



Obr. 3 Diagram algoritmu