SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

LOKALIZÁCIA A NAVIGÁCIA MOBILNÉHO ROBOTA

SEMINÁRNA PRÁCA

2024 Bc. Eduard Zelenay, Bc. Maroškokh Kocúr

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

LOKALIZÁCIA A NAVIGÁCIA MOBILNÉHO ROBOTA

SEMINÁRNA PRÁCA

Študijný program:	Robotika a kybernetika
Predmet:	I–RMR – Riadenie mobilných robotov
Prednášajúci:	prof. Ing. František Duchoň, PhD.
Cvičiaci:	Ing. Martin Dekan, PhD.

Bratislava 2024 **Bc. Eduard Zelenay, Bc. Maroškokh Kocúr**

Obsah

Zoznam obrázkov a tabuliek

Zoznam algoritmov

Zoznam výpisov

algorithm algpseudocode algorithm2e

Úvod

V prvom zadaní je našou úlohou vytvoriť aplikáciu, ktorá bude zobrazovať aktuálny stav robota pomocou tele riadenia a popri tom bude spĺňať základné normy pre HMI. Tato aplikácia bude slúžiť ako nástroj pre ovládanie robota a zobrazovanie jeho stavu.

Toto zadanie sa bude zaoberať tromi hlavnými problematikami:

- Návrh aplikácie - v tejto časti sa budeme venovať návrhu aplikácie, respektíve opraveniu štruktúry už existujúcemu grafickému rozhraniu.
- Fúzia kamery a lidaru - druha sekcia sa zaoberá fúziou dát z kamery a lidaru.
- Vizuálna detekcia - posledná sekcia sa zaoberá tele riadením robota.

1 Zadanie

Cieľom predmetu je naučiť sa základy riadenia pohybu mobilných robotov najmä z hľadiska senzorového vybavenia a k tomu určených algoritmov lokalizácie, navigácie, mapovania a plánovania pohybu robota v prostredí. Úlohy budete programovať na počítači, ktorý sa bude pripájať na diaľku k mikropočítaču Raspberry Pi, ktoré je umiestnené na každom robote. Raspberry Pi preposiela údaje z robota a laserového diaľkomeru (v ďalšom texte sa môže objaviť označenie lidar, alebo len skrátene laser.) pomocou protokolu UDP.

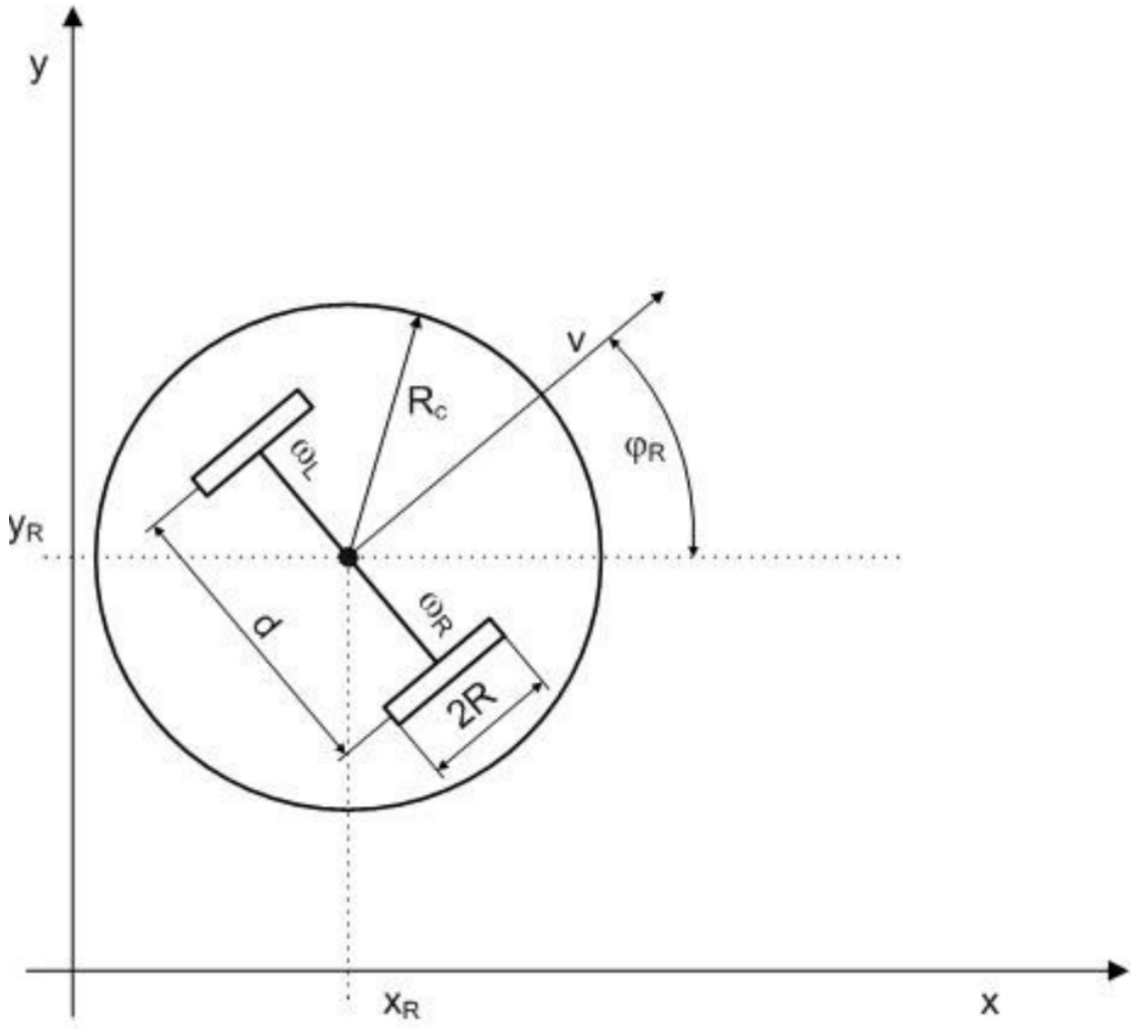
Na počítačoch v laboratóriu (alebo na https://github.com/dekdekan/demoRMR-all ak budete robiť na vlastnom notebooku) sa nachádza demo program, ktorý má v sebe urobenú základnú komunikáciu ako s robotom tak aj s lidarom. Demo program je spravený vo vývojovom prostredí QT a jazyku C++. Ak vám tento jazyk/prostredie nevyhovuje, môžete robiť v čomkoľvek inom, ale komunikáciu a parsovanie dát si musíte spraviť samy. Rovnako nie je nutné aby váš výsledný program vyzeral ako tento demo program, bez problémov akceptujeme ako výsledok konzolovú aplikáciu.

Na cvičeniach budete mať za úlohu vypracovať projekt pozostávajúci z čiastkových úloh rozdelených do štyroch blokov. Z každého bloku si môžete vybrať buď úlohu, ktorá je vysvetlená v tomto manuáli (a bude podrobnejšie prejdená aj na cvičení) alebo niektorú z metód vysvetlených na prednáškach (s bodovým ohodnotením podľa náročnosti).

Celkovo je možné na cvičeniach získať 35 bodov za úlohy a maximálne 5 bodov za súťaž. Body budú pridelené na základe predvedenia fungovania algoritmu a odovzdanej dokumentácie. Dokumentácia musí obsahovať užívateľskú príručku (informácie o tom, ako program spustiť, čo robia jednotlivé gombíky/ aké príkazy váš program podporuje) a popis fungovania jednotlivých algoritmov (slovne, pseudo kód, vývojový diagram)

1.1 Súťaž

Súťaž prebehne na konci semestra. Podmienkou účasti je schopnosť presunúť robota z bodu A do bodu B (bez prekážok). Cieľom súťaže je dostať robota z bodu A do bodu B v priestore s prekážkami. Roboty sa budú pohybovať v čiastočne známej mape, t.j. dostanete neúplnú mapu prostredia. Pri súťaži sa vyhodnocuje čas, za ktorý sa dostanete do cieľa a presnosť s akou sa dostanete do bodu B. Najlepší získajú 5b najhorší 3b. Pre účastníkov ktorý nedôjdu do cieľa ale ich robot sa vyhol aspoň jednej prekážke sú rezervované 2 body. Za účasť je 1 bod.



Obr. 1.1: Kinematická schéma diferenciálneho podvozku mobilného robota

1.2 Úloha 1

Úlohou lokalizácie je povedať, kde v priestore sa mobilný robot nachádza. Najjednoduchší spôsob lokalizácie mobilného kolesového robota s diferenciálnym podvozkom (t.j. taký ako sapoužíva na cvičeniach) je odometria. Odometria je typ relatívnej lokalizácie (určuje polohu voči predchádzajúcej polohe), ktorá prírastok polohy určuje na základe otočenia kolies. Keďže robot Kobuki má v sebe aj gyroskop môžete natočenie robota spresniť a vytvoriť takzvanú gyroodometriu. Na výpočet odometrie pre robot zobrazený na obrázku 1, je nutné aplikovať tieto vzorce:

Úlohou polohovania je dostať mobilný robot na želané súradnice. To je pre robot Kobuki možné dosiahnuť riadením rýchlosti jednotlivých kolies. Na to, aby ste dosiahli želanú polohu presne, je nutné navrhnúť regulátor (P,PI,PID). Najjednoduchšie riešenie poloho-

vania, je rozdeliť pohyb robota na transláciu a rotáciu (t.j. robot sa buď hýbe dopredu alebo sa točí na mieste) a pre každý z týchto pohybov navrhnúť regulátor samostatne. Pseudo-kód diskrétneho regulátora vyzerá takto:

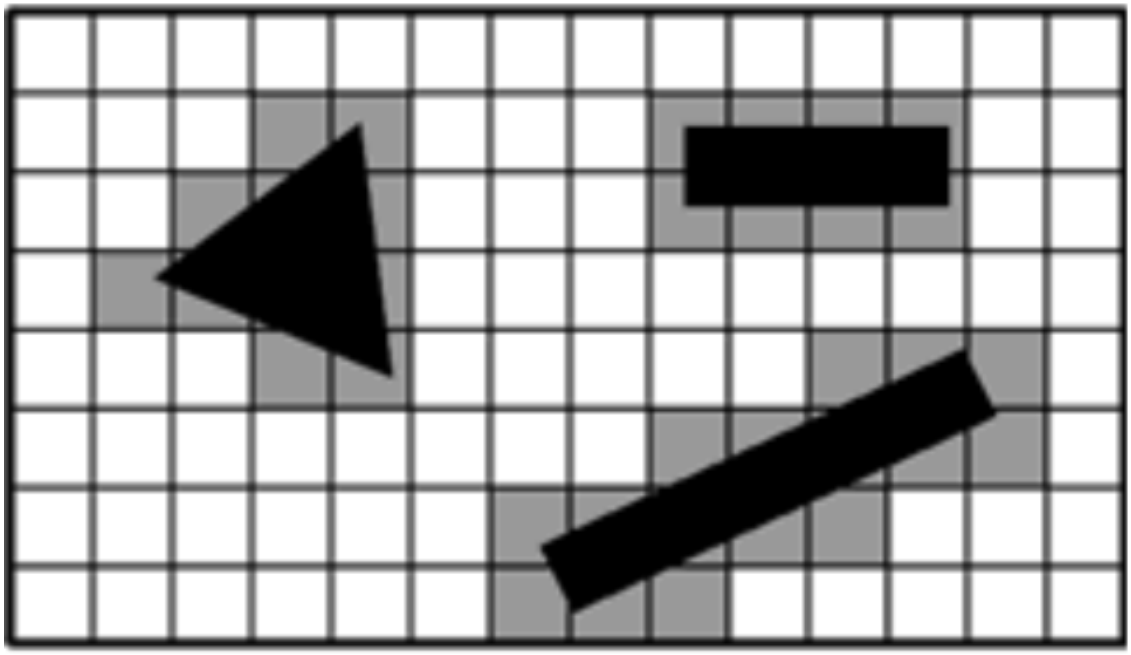
1.3 Úloha 2

Navigácia mobilného robota zabezpečuje bezkolízny prechod robota prostredím. Na svoju činnosť využíva aktuálne údaje zo snímačov a nepotrebuje poznať prostredie vopred (mapu). K najjednoduchším algoritmom navigácie patria hmyzie algoritmy (bug algorithm), z toho len niektoré na svoje fungovanie využívajú laserový diaľkomer. Takýmto algoritmom je hmyzí algoritmus - typ dotyčnica (tangent bug algorithm). Princíp algoritmu je:

- Ak sa dá ísť na cieľ a nie si v móde sledovanie steny, choď na cieľ.
- Ak je v ceste prekážka a nie si v móde sledovania steny, tak smeruj na jej hranu tak, aby bola euklidovská vzdialenosť, ktorú má robot prejsť čo najmenšia. Zapamätaj si aktuálnu vzdialenosť do cieľa.
- Ak sa niekedy vzdialenosť od cieľa začne zväčšovať prepni sa na mód sledovanie steny.
- Ak sleduješ stenu, tak ju sleduj až do prípadu až kým vzdialenosť do cieľa nie je menšia ako najkratšia zapamätaná vzdialenosť do cieľa.
- Zastav, ak si dosiahol cieľ .

1.4 Úloha 3

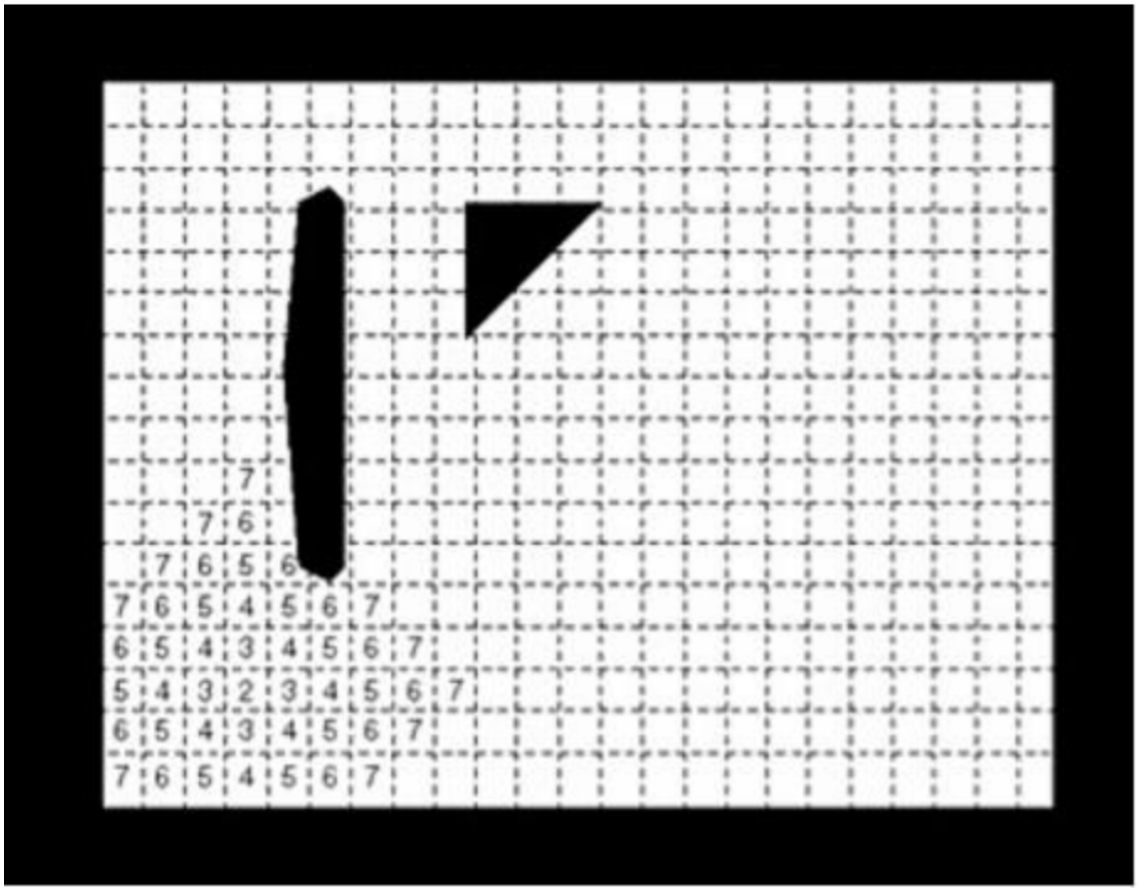
Úlohou mapovania je vytvoriť reprezentáciu prostredia na základe údajov zo snímačov robota. Inak povedané, mapovanie je fúzia (spájanie) informácií o polohe robota s informáciami o prekážkach získaných z laserového diaľkomera. Najjednoduchšia reprezentácia prostredia je mriežka obsadenia. Mriežka obsadenia rozdelí priestor na konečný počet prvkov (buniek), kde každý prvok obsahuje informáciu o jeho obsadenosti (t.j či sa tam nachádza prekážka). Prerátanie údajov získaných z laserového diaľkomera do súradníc robota je možné na základe homogénnej transformácie. V prípade, že zanedbáme pohyb v osi Z (t.j hýbeme sa v rovine), ako aj rotácie okolo osi X a Y (robot sa pri pohybe nenakláňa do žiadneho smeru) je možné použiť homogénnu transformáciu pre 2 rozmery. 2D transformáciu je možné zapísať ako:



1.5 Úloha 4

Úlohou plánovania dráhy je vygenerovať optimálnu trasu pre mobilný robot na základe existujúcej mapy prostredia. Na tento účel sa využije typ mapy z predchádzajúcej úlohy a použitý algoritmus na plánovanie dráhy je záplavový algoritmus (flood fill algorithm). Jeho princíp je nasledovný:

- Cieľová bunka je ohodnotená na 2. Všetky bunky susediace s touto bunkou sú ohodnotené na 3.
- Nulové bunky susediace s bunkami s hodnotami 3 sú ohodnotené na 4.
- Procedúra pokračuje dovtedy, kým nie je spojený štart a cieľ.


7
7 6
7 6 5 6
7 6 5 4 5 6 7
6 5 4 3 4 5 6 7
5 4 3 2 3 4 5 6 7
6 5 4 3 4 5 6 7
7 6 5 4 5 6 7

2 Odometria

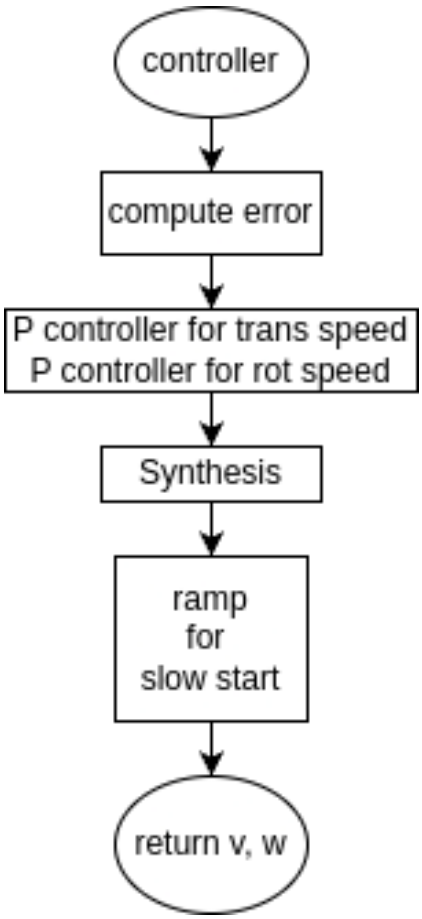
Odometriu sme riešili pomocou výstupov z enkóderov z motorov a gyroscopu pripevnenom na robotovi.

Na určenie polohy [X,Y] sme použili inkrementálne enkódre a uhol natočenia θ sme určovali pomocou gyroskopu.

Na začiatku sme vyčítali dáta zo senzorov, ktoré sme neskôr odčítavali od vyčítaných hodnôt, čo zabezpečil, že poloha robota sa určovala od súradnicového systému pri zapnutí robota [0,0,0].

3 Riadenie

Zvolili sme si P regulátor, na riadenie polohy robota. Aby robot nabiehal plynulo na žiadanú rýchlosť implementovali sme rampu ako rozbehový člen, kde každým zavolaním regulátora, ktorý sa vola periodicky zvýšime rýchlosť 5% pokiaľ sa nedostane na maximálnu rýchlosť.



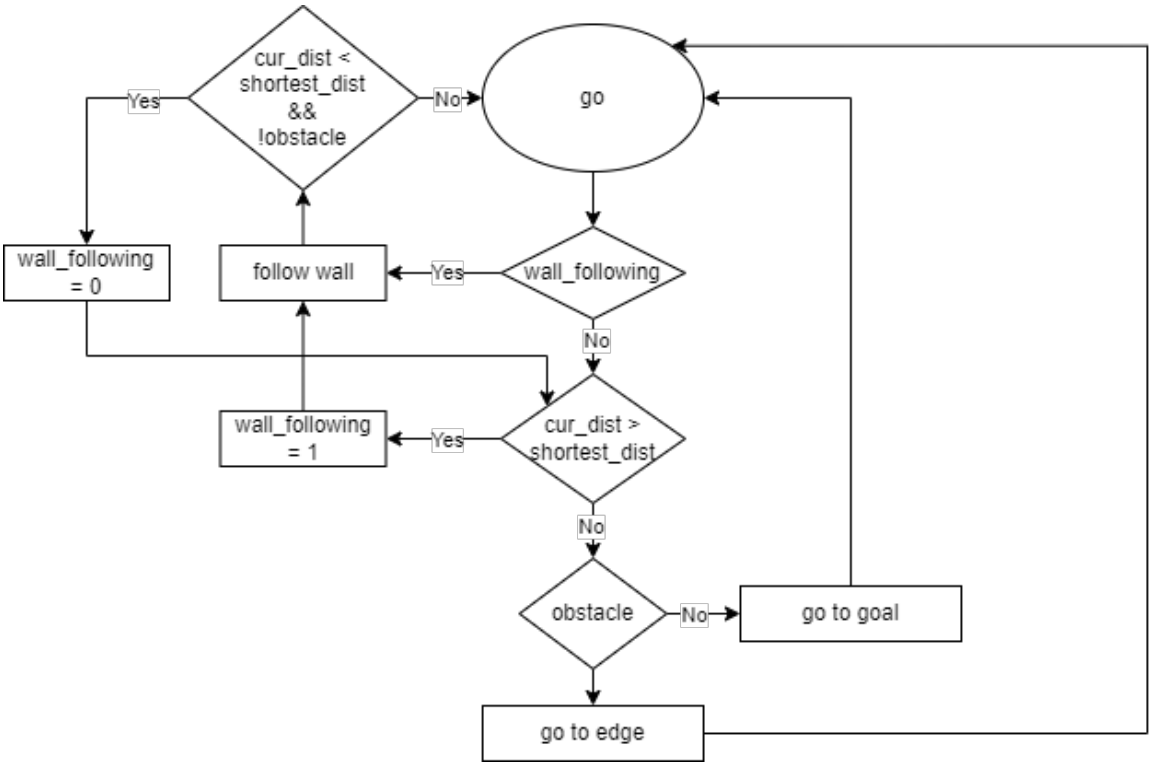
Obr. 3.1: Blokový diagram regulátora

Výstupom regulátora je translačná a rotačná rýchlosť, sktorú následne prepočítame na žiadaný pohyb robota (translačna, rotácia na mieste alebo pohyb po kružnici).

Parametre regulátorov: rýchlosti $P = 6$ a rotácie $P = 2$. Prírastok rampy je 5%.

4 Obchádzanie prekážok

Obchádzanie prekážok, alebo reaktívna navigácia by sa dala rozložiť do dvoch menších častí. Časť obchádzania objektu hmyzím algoritmom a časť sledovania steny. Zjednodušený pohľad na celkový algoritmus obchádzania prekážok môžeme vidieť na 4.1.

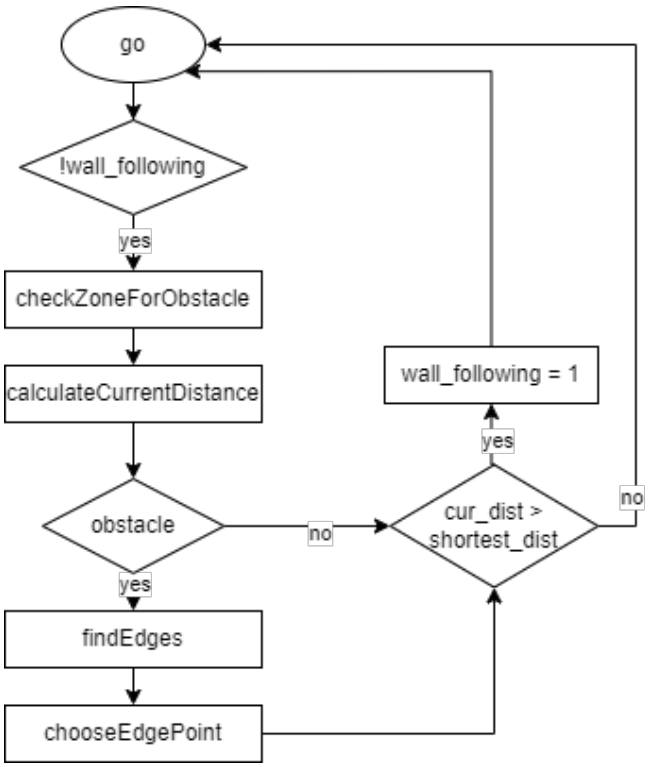


Obr. 4.1: Blokový diagram obchádzania prekážok

Ak robot nie je v móde sledovania steny a v ceste nie je prekážka, robot ide na cieľ. Ak sa kedykoľvek počas cesty začneme vzďalovať od prekážky, prepne sa na mód sledovania steny. Ak je v ceste prekážka robot ide na hranu tejto prekážky.

4.1 Hmyzí algoritmus

Prvá časť algoritmu spočíva v obchádzaní prekážky pomocou hmyzieho algoritmu. Zjednodušenú formu algoritmu môžeme vidieť na 4.2.



Obr. 4.2: Blokový diagram hmyzieho algoritmu

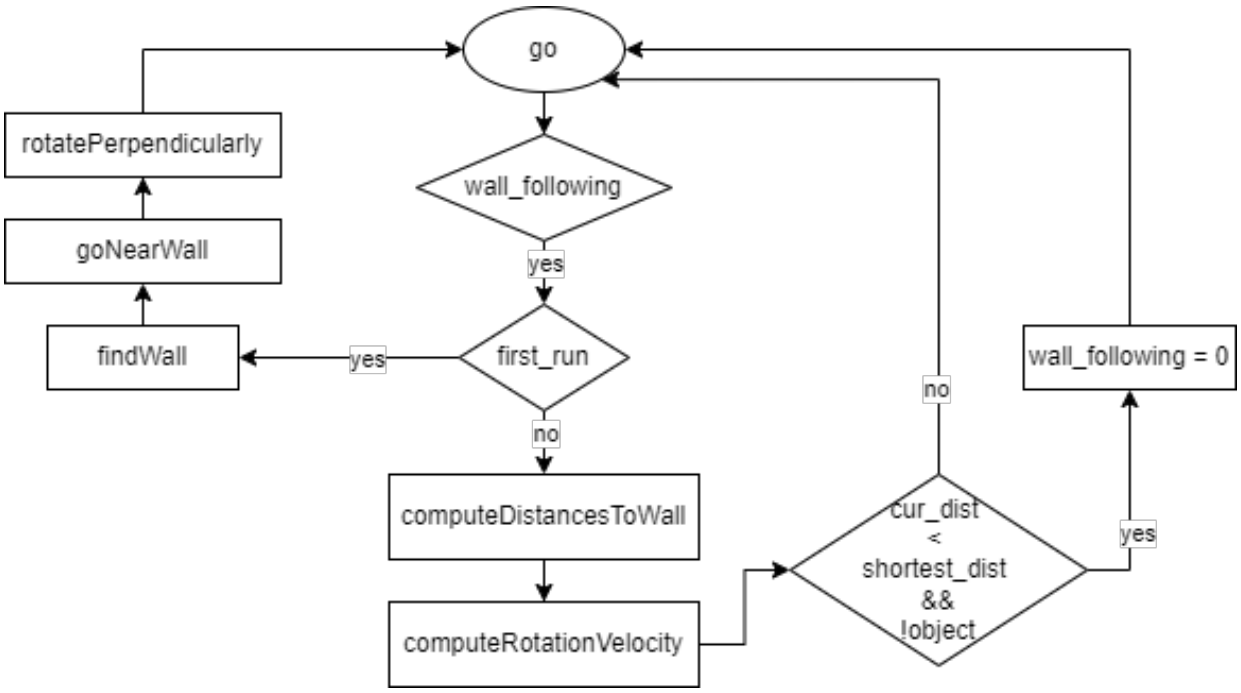
Ak robot nie je v móde sledovania steny, skontroluje zónu, ktorá smeruje do cieľa, zistí či sa niekde v nej nachádza prekážka. Následne sa vypočíta aktuálna vzdialenosť od cieľa.

Ak sa v zóne nachádza prekážka, nájde jej rohy. V prípade, že by rohy nenašiel, začne sledovať stenu. Ak našiel oba rohy, odsadí od nich rohové body, vypočíta vzdialenosti do cieľa skrz tieto rohové body a vyberie si ten s kratšou z nich. V prípade, že nájde iba jeden rohový bod, vyberie si ten.

Ak sa v akejkoľvek chvíli začne robot vzďalovať od cieľa, prepne sa do módu sledovania steny.

4.2 Sledovanie steny

Druhá časť algoritmu obchádzania prekážok je algoritmus sledovania steny. Jeho zjednodušenú podobu môžeme vidieť na 4.3.



Obr. 4.3: Blokový diagram sledovania steny

Ak je robot v móde sledovania steny, overí sa, či je pripravený na jej sledovanie. Ak ešte nie je, nájde sa stena najbližšie ku nemu, robot sa presunie ku stene a natočí sa kolmo na ňu, buď doľava, alebo doprava, podĺa toho, kde sa stena nachádzala relatívne od natočenia robota.

Ak už je pripravený na sledovanie steny, vypočítajú sa vzdialenosti od steny z ľavej, pravej strany a spred robota. Na základe určenej vzdialenosti od steny sa zistí, na ktorej strane je robot bližšie ku stene, ako je dané. Podľa toho sa vypočíta rotačná rýchlosť pre robot, ktorý sa neustále hýbe po krivke s danou translačnou rýchlosťou. Z rotačnej rýchlosti sa vypočíta polomer, ktorý má krivka opisovať.

Ak je vzdialenosť menšia ako kedykoľvek bola a zároveň má robot voľnú cestu na cieľ, vypne sa sledovanie steny.

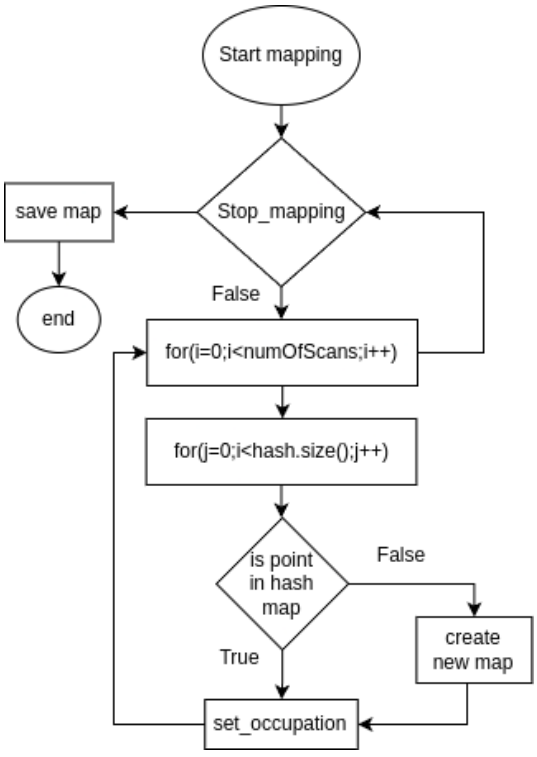
5 Mapovanie

Na mapovanie sme si vytvorili v užívateľskom okne tlačítko, ktorým sme spúšťali mapovanie. Na mapovanie je možne použiť manuálne ovládanie, alebo robotovi prideliť body cez ktoré ma prejsť.

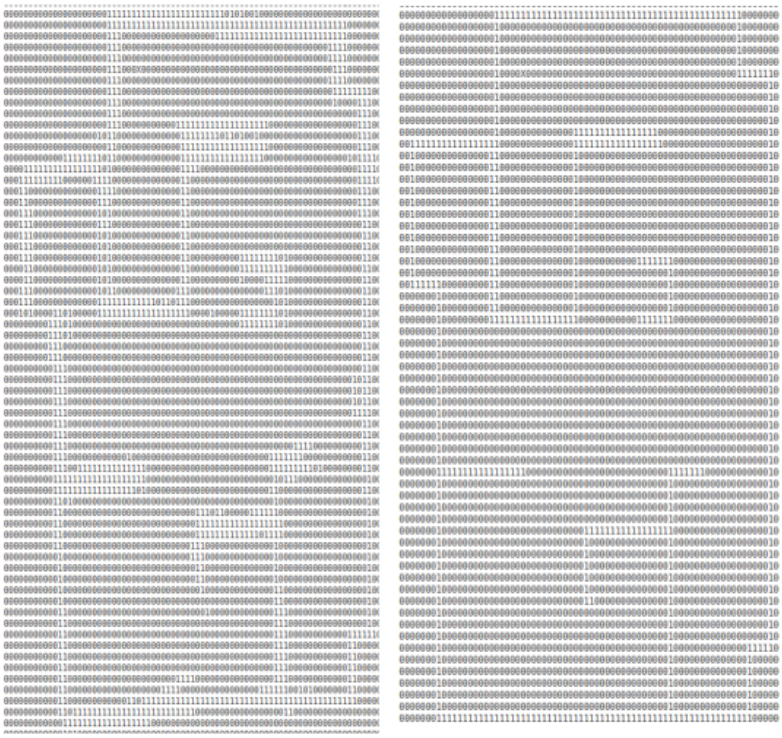
Na začiatku si vytvoríme hash mapu o veľkosti 60 štvorčekov s rozmermi 10x10cm, stred počiatočnej hash mapy je poloha robota.

Počas mapovania prechádzame dáta z Lidaru a zisťujeme, či sa nachádzajú už vo vytvorenej hash mape. Ak je bod mimo vytvorí sa nova mapa posunutá stredom o veľkosť hash mapy.

Po ukončení mapovania zistíme krajné body mapy, vďaka ktorým vieme určiť veľkosť mapy. Mapu uložíme do textového súbora, v ktorom sa nachádzajú body opisujúce priestor [x,y] a v hlavičke súboru sa nachádza stred výslednej mapy a jej veľkosť.



Obr. 5.1: Blokový diagram mapovania

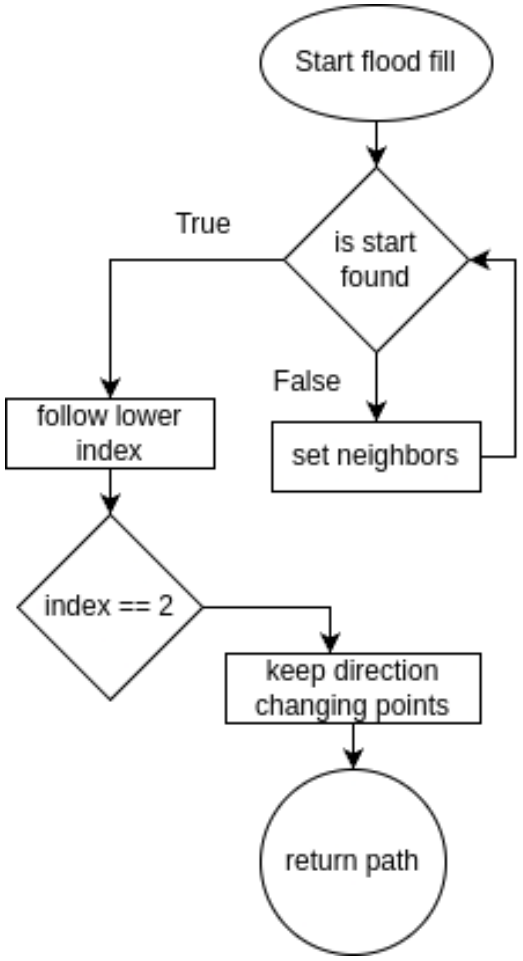


Obr. 5.2: Porovnanie mapovania realnej(vľavo) oproti simulácii(vpravo)

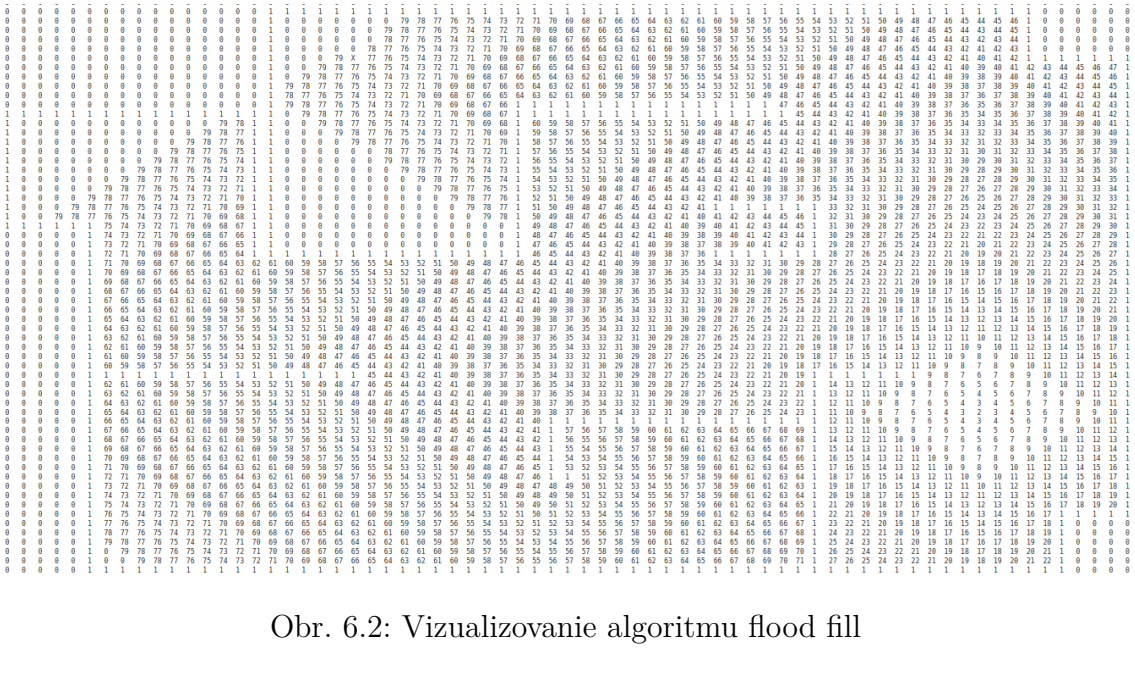
6 Hľadanie najkratšej trajektórie

Na najdenie najkratšej trajektórie sme si zvolili Flood fill algoritmus. Algoritmus funguje tak, že cieľovému bodu nastavíme index 2 a potom na základe štvorsusednosti, prejdeme každy bod a nastavíme mu index o jedna väčší ako susedovi s najväčším indexom. Algoritmus sa ukončí ak prejdeme všetky políčka v uzavretom priestore alebo ak narazíme na cieľ.

Po vyplnení mapy indexami, hľadame najkratšiu vzdialenosť od štartu do cieľa. Zvolili sme si priority aby algoritmus priorizoval ľavu, prednú pravú a zadnú stranu. Algoritmus pozera do svojho okolia a hľadá suseda s menším indexom ako ma on sám. Výsledkom sú body cez ktoré by mal robot prejsť. Aby to bolo použiteľne pre náš regulátor, vymazali sme prejazdové body a nechali sme iba body, kde robot mení smer.



Obr. 6.1: Blokový diagram flood fill algoritmu



Obr. 6.2: Vizualizovanie algoritmu flood fill

Záver

Podarilo sa nám navrhnúť riadenie robota, ktoré sme testovali na reálnom robote s názvom Kobuki. Odometria a riadenie sme využili pri riešení mapovania, kde sme zistili nedostatky odometrie na reálnom robote. Ako môžeme vidieť na 5.2 tak reálna mapa oproti simulácii mala menšie odchýlky, spôsobené nepresnosťou otáčania robota. Všetky úlohy sa nám podarilo otestovať na súťaži, kde sme súťažili proti spolužiakom a podarilo sa nám získať fantastické výsledky, vďaka ktorým sme vyhrali Limitovanú edíciu knihy Riadenie mobilných robotov, ktorú napísal Pán prof. Ing. František Duchoň, PhD. a kolektív.