Ústav robotiky a kybernetiky, Fakulta elektrotechniky a informatiky
Slovenskej technickej univerzity, Ilkovičová ulica č.3, 812 19 Bratislava

Riadenie mobilných robotov

Riadenie robota Kobuki

Bratislava
2023

riešiteľ:
Bc. Leonard Haraslín
Bc. Martin Berki

Zadanie:

1. Úloha Lokalizácia a polohovanie robota v prostredí
2. Úloha Navigácia
3. Úloha Mapovanie
4. Úloha Plánovanie trajektórie

1.)

Pri riešení danej úlohy sme sa zamerali na implementáciu v súbore mainwindow.cpp v rámci funkcie process:ThisRobot. V tejto funkcii sme si vytvorili potrebné premenné, vrátane premenných pre enkóder.

Z enkódera sme potrebovali získať diferenciu medzi aktuálnou a predchádzajúcou hodnotou enkódera. Túto hodnotu sme následne normalizovali pomocou nasledujúceho vzťahu:

```
int Odometry::normalizeDiff(int diff)
{
    if (diff > SHRT_MAX) return diff - USHRT_MAX;

    else if (diff < SHRT_MIN) return diff + USHRT_MAX;

    else return diff;
}
```

Využili sme princíp odometrie na lokalizáciu robota, ktorý pozostával z nasledujúcich vzorcov:

$$x_{k+1} = x_k + l_k \cos \varphi_k$$

$$y_{k+1} = y_k + l_k \, sin \, \varphi_k$$

$$\varphi_{k+1} = \varphi_k + \Delta\alpha$$

$$\Delta\alpha = \frac{l_{kr} - l_{kl}}{d}$$

V tejto úlohe sme tiež potrebovali normalizovať uhly, čo sme dosiahli pomocou nasledujúcej funkcie:

```
double Control::normalizeAngleError(double error)
{

    if(error > M_PI) return -2*M_PI + error;

    if(error < -M_PI) return +2*M_PI + error;

    return error;

}
```

Tieto výpočty a normalizácie nám umožnili správne lokalizovať robota a vyhodnocovať jeho pohyb na základe enkóderových hodnôt.

2.)

V druhej úlohe sa zameriavame na bezkolízny prechod robota prostredím, kde na svoju činnosť využíva aktuálne údaje zo snímačov.

Pri riešení danej úlohy sme uplatnili postup, ktorý spočíval v získavaní hodnôt senzorového lúča a následnom spracovaní týchto hodnôt s cieľom získať spoľahlivé informácie o prostredí. Okrem toho sme vyvinuli navigačnú stratégiu, ktorá nám umožnila efektívne riadiť pohyb robota a minimalizovať kolízie s prekážkami. Pokiaľ sme chceli vyhodnotiť jednu vzorku senzorového lúča, zbierali sme sériu 10 hodnôt z toho istého smeru. Tieto hodnoty sme následne zpracovali výpočtom priemeru, čím sme získali reprezentatívnu hodnotu pre daný smer lúča. Tento postup nám pomáhal eliminovať prípadné náhodné šumy alebo odchýlky v meraniach senzorov.

V súvislosti s navigáciou sme sa zamerali na detekciu kolízií a následné riadenie robota tak, aby sa vyhol prekážkam a pohyboval sa požadovaným smerom. Keď robot detekoval kolíziu, znamenalo to, že sa nachádza blízko prekážky. V takom prípade sme aktivizovali procedúru narovnávania robota smerom pozdĺž steny.

Po detekcii kolízie sa robot postupne približoval k stene a monitoroval sme zmeny v uhle senzorového lúča. Pokračovali sme v narovnávaní, kým sme zistili, že nasledujúci uhol senzorového lúča je väčší ako predošlý. Toto nám umožnilo identifikovať miesto, kde robot opustil stenu a mal by pokračovať vo svojom pohybe voľným priestorom.

Týmto spôsobom sme dosiahli, že robot sa vyhol kolíziám s prekážkami tým, že sledoval priebeh steny a následne narovnával smer pohybu, keď uhol senzorového lúča signalizoval, že sa nachádzame mimo stenu. Táto navigačná stratégia nám umožnila presnejšie a efektívnejšie riadiť pohyb robota v prostredí s prekážkami.

3.)

Pri riešení danej úlohy sme sa zamerali na spracovanie dát z lidaru. Postupovali sme nasledovne:

Získanie hodnôt z lidaru: Prešli sme do súboru mapping.cpp, kde sme využili príslušnú funkciu na získanie dát z lidaru. V tejto funkcii sme pristupovali k štruktúre, ktorá obsahovala potrebné dáta zo snímaných bodov pomocou lidaru.

```
if(sd > 130 && sd < 3000 && !(640> sd && sd > 700))
```

Normalizácia jednotiek: Aby sme zabezpečili súlad medzi jednotkami v našej koordinátovom systéme a dátami z lidaru, bolo potrebné vykonať úpravu jednotiek. Naše koordináty sme vynásobili 100 a hodnoty z lidaru sme vydelili 10 aby boli v rovnakom meradle.

```
double x = coords[0]*100 + laserData->Data[i].scanDistance * Odometry::cosd(coords[2]*180/M_PI - laserData->Data[i].scanAngle)/10;
double y = coords[1]*100 + laserData->Data[i].scanDistance * Odometry::sind(coords[2]*180/M_PI - laserData->Data[i].scanAngle)/10;
```

Vytvorenie mapy koordinátov: Po normalizácii sme hodnoty x a y vydelili 10 a zaokrúhlili sme ich na najbližšie celé číslo. Tým sme získali mapu súradníc, ktorú sme následne použili na vytvorenie mriežky (grid).

Transformácia hodnôt na mapu jednotiek a núl: Na základe mapy koordinátov sme vykonali transformáciu hodnôt na mapu, kde jednotka reprezentovala dostupný priestor a nula reprezentovala prekážky. Táto transformácia nám umožnila získať vizuálnu reprezentáciu prostredia a identifikovať dostupné a nedostupné oblasti pre robota.

Týmto spôsobom sme úspešne namapovali dáta z lidaru, vykonali potrebné úpravy jednotiek a transformovali ich na mapu, kde sme vytvorili mriežku s hodnotami jednotiek a núl.

4.)

Pri riešení nasledujúcej úlohy sme použili metódu nazývanú "flood fill algoritmus" na mapovanie mriežky. Úlohou bolo preskúmať priestor a určiť dostupné a nedostupné oblasti pre robota. Pred začiatkom samotného algoritmu sme zistili, že bolo potrebné zväčšiť prekážky v mriežke. Rozhodli sme sa ich rozšíriť o 3 jednotky v smere y-osi a o 2 jednotky v smere x-osi. Toto rozšírenie bolo nevyhnutné na správne určenie dostupnosti priestoru pre robota.

Flood fill algoritmus je postup, pri ktorom sa vyplní uzavretý priestor zvoleným začiatočným bodom. V našom prípade sme vyplňovali mriežku, pričom sme využili diagonálne pohyby robota.

Na začiatku sme určili začiatočný bod v mriežke, kde sa robot nachádza. Následne sme šírili vlnu z tohto bodu do okolitých voľných buniek, pričom sme využívali diagonálne smerovanie. Vlna sa šírila, pokiaľ sme narazili na prekážku alebo sme vyplnili celú dostupnú oblasť. Pri každom kroku sme zaznamenávali informácie o dosiahnutej bunke, ako napríklad jej pozíciu, vzdialenosť od začiatočného bodu a informáciu o tom, či je táto bunka dosiahnuteľná alebo nedosiahnuteľná.

Po dokončení algoritmu sme mali detailnú mapu mriežky s označením dostupných a nedostupných oblastí pre robota. Táto mapa nám poskytla presný prehľad o priestore, ktorý je pre robota prístupný. Na základe tejto mapy sme potom mohli plánovať a rozhodovať o optimálnych pohyboch robota na dosiahnutie cieľa. Použitie flood fill algoritmu a zväčšenie prekážok nám umožnilo efektívne a presne určiť dostupnosť priestoru pre robota. Tým sme získali dôležité informácie pre ďalšie rozhodovanie a plánovanie pohybu robota v rámci danej úlohy.