LAR
Semestrální práce - Parkování robota

Lukáš Navara, Josef Macháč, Jan Koča

I. Úvod

Tento dokument je technickou zprávou k semestrální práci v předmětu LAR, jenž je vyučován ve 4. či v 6. semestru oboru Kybernetika a robotika na Fakultě elektrotechnické ČVUT.

II. Zadání

Robot TurtleBot má za úkol samostatně zaparkovat v garáži tvořené kartonovými krabicemi (kvádry). Zadání je rozděleno do tří úloh podle náročnosti řešení:

1) Robot zaparkuje v garáži, pokud vidí vjezd.
2) Robot lokalizuje garáž a zaparkuje v ní.
3) Robot zaparkuje v garáži v prostředí s překážkami.

Pro podrobnější specifikace zadání vizte zadání úlohy na stránkách cvičení. Odkaz naleznete v sekci XIII.

III. Rozbor zadání a specifikace problému

Zadání ponechává prostor pro vlastní interpretaci a doplnění problému. Zároveň je možné zvolit jednu ze 3 nabízených variant dle obtížnosti. Úkol jsme tedy blíže specifikovali následovně.

Vybrali jsme 3. variantu zadáni. Robot se bude snažit zaparkovat do obecně orientované garáže, jež se nachází někde v jeho okolí, v prostředí s překážkami.

Na základě těchto informací konkretizujeme zadání takto:

- Řešení rozdělíme do dvou základních fází:
    1) **Robot se dostane před bránu garáže**: Robot po stisknutí tlačítka analyzuje své okolí; nalezne překážky, garáž a její vjezd. Zanese tyto informace do mapy, ve které posléze nalezne vhodnou cestu, podle které pojede před vjezd garáže tak, že se vyhne překážkám po cestě. Mapa bude vždy vykreslena pouze pro aktuální pohled robota, tzn. nebudou v ní zaznamenávány objekty mimo zorné pole kamery.
    2) **Robot zaparkuje do garáže**: Robot se bude nacházet před vjezdem do garáže. Bude již v takové blízkosti, že mezi ním a vjezdem do garáže nebude žádná překážka. Vycentruje se před vjezd garáže a vjede dovnitř tak, aby nenarazil do stěn. Konec této operace signalizuje zvukovým signálem po zastavení.

Ze zadání dále známe barvy jednotlivých objektů uvažovaných při řešení úkolu. Překážkové sloupky mají červenou, zelenou, nebo modrou barvu. Garáž má barvu žlutou až na stěny u vjezdu, které jsou fialové.

IV. Robot TurtleBot

Robot TurtleBot, kterého máme pro řešení problému k dispozici, se nachází v Praze, konkrétně v prostorách FEL na Karlově náměstí v budově E.

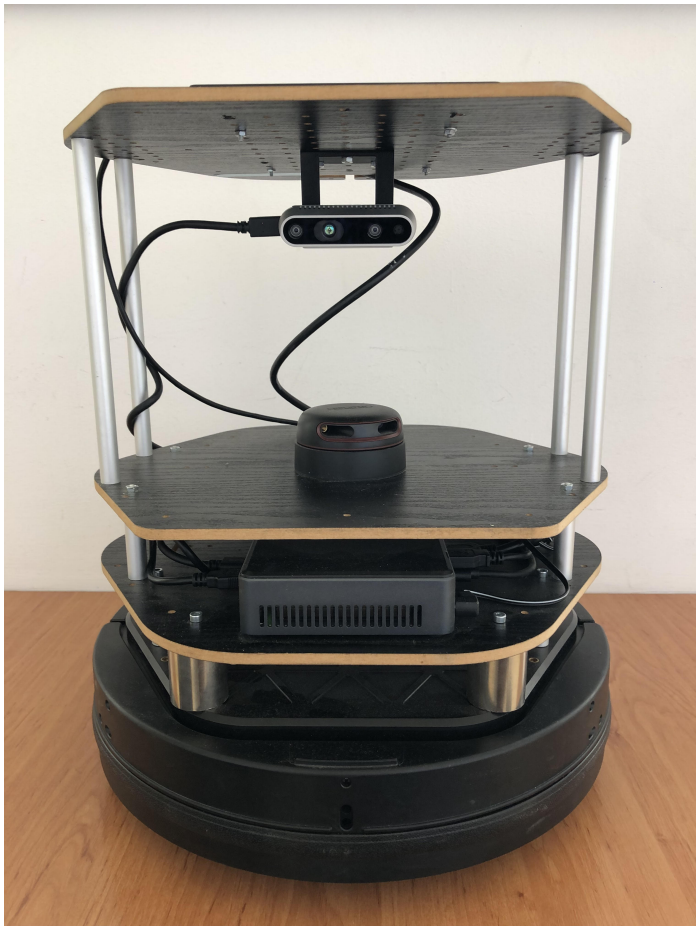

Fig. 1: Robot TurtleBot

Dále je dispozici několik druhů kamer. Při implementaci řešení úlohy jsme využívali 3D kameru RealSense D435.

Bližší specifikace robota lze najít v sekci XIII.

V. Zpracování RGB obrázku

V této sekci představíme postupy a hlavní myšlenky stojící za zpracováním obrazu pro naši úlohu. Více o tom, jak jsme konkrétně tyto postupy implementovali, v sekci IX-B.

A. Definování barev pomocí HSV prahů

K tomu, abychom nalezli překážky, garáž a sloupky vjezdu na obrázku, využijeme faktu, že každý z těchto objektů má

předem definovanou barvu jednoduše rozlišitelnou od ostatních.

Každý pixel na obrázku má RGB hodnotu, kterou převedeme na hodnotu HSV. Experimentálně jsme zjistili HSV prahy pro jednotlivé barvy relevantních objektů.

Například pro zelenou překážku jsem určili z několika obrázků, na kterých jsme zachytili její možné orientace a pozice, spodní a horní HSV prahy, kterými jeho barvu definujeme pro naše řešení. Padne-li tedy HSV hodnota pixelu do takto zjištěného intervalu, rozhodneme, že pixel náleží zelené překážce.

B. Detekce objektů

RGB obrázek, který chceme zpracovat, převedeme do HSV formátu. Pro každou barvu, kterou má některý z potenciálních objektů, které se mohou objevit na obrázku, využijeme zjištěných HSV prahů k segmentaci.

Hledání objektů předvedeme na konkrétním případě, vizte Fig. 2. Pro každou z barev provedeme na obrázku stejné operace. Popíšeme tedy obecný postup pro některou z nich.

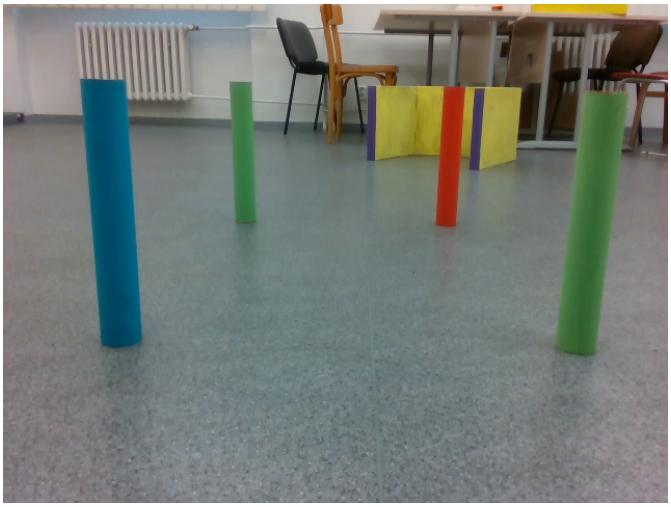

Fig. 2: RGB obrázek a možné rozložení objektů zájmu na něm

Nejdříve pro HSV obrázek vytvoříme pomocí HSV prahů masku. Maska je stejné velikosti jako původní obrázek. Je binární a to takovým způsobem, že pixely, které svou HSV hodnotou padly mezi zjištěné prahy, mají hodnotu jedna. V opačném případě nesou hodnotu nula. Pro příklad vizte Fig. 3.

Fig. 3: Binární masky pro zelenou (vlevo) a žlutou (vpravo) barvu

Na takto získaných maskách nalezneme kontury, které jsou v tomto případě hranicemi objektů dané barvy na obrázku.

Dále zkontrolujeme obsah nalezených kontur a budeme pokračovat pouze s těmi, které obsahují alespoň 300 pixelů. Tato hodnota byla experimentálně zjištěna jako vhodná. Tím se zbavíme případných nepřesností způsobených například nežádoucím odrazem světla při segmentaci či příliš vzdálených objektů.

Pro žlutou barvu, tzn. garáž, již kontury nebudeme dále zpracovávat. U ostatních barev zbylé kontury uzavřeme do nejmenšího možného obdelníku, neboť s obdelníkem se lépe manipuluje a lze jej úsporněji reprezentovat (např. 4 body), který následně o něco zmenšíme. Důvod pro tuto operaci a více informací zmíníme v části VI.

Pro ilustraci tyto kontury a zmenšené ohraničující obdelníky zakreslíme do původního obrázku. Výsledek vizte na Fig. 4.

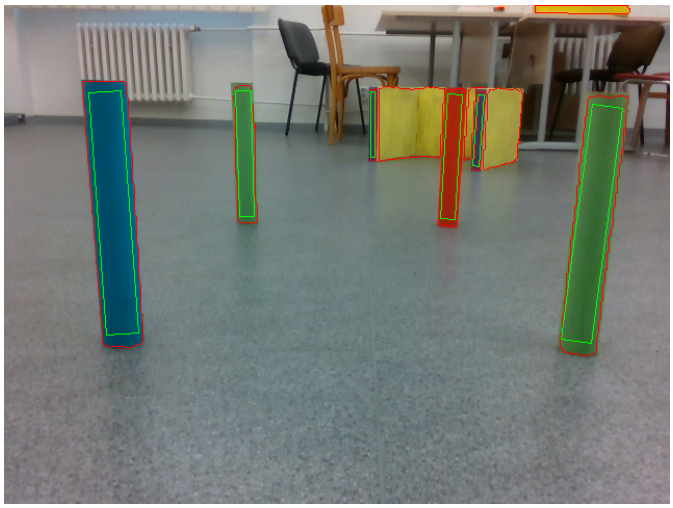

Fig. 4: RGB obrázek a detekované objekty, nalezené kontury - červená, zmenšené ohraničující obdelníky - zelená

VI. ZPRACOVÁNÍ POINT CLOUDU

K řešení úlohy máme k dispozici také 3D kameru, která nám poskytuje prostorová data ve formě point cloudu.

Point cloud v našem případě přiřazuje každému pixelu na RGB obrázku souřadnici (x, y, z) v prostoru. Bod (0, 0, 0) odpovídá kameře. Souřadnice jsou udávány v metrech.

Před použitím upravíme jeho formát, aby byla práce s ním intuitivnější. Chceme, aby kladná část osy x mířila kolmo doprava od kamery, kladná část osy y mířila kolmo před kameru a kladná část osy z mířila kolmo nad kameru.

V sekci III jsme avizovali, že budeme zakreslovat nalezené objekty do 2D mapy. Kdyby byla kamera orientována tak, aby osy x a y byly rovnoběžné s podlahou, zajímaly by nás (jakožto jejich 2D projekce do roviny podlahy) pouze souřadnice (x, y).

Kamera je však nainstalována tak, že aby byla tato podmínka splněna, musíme všechny body orotovat kolem osy x o experimentálně zjištěných -13,5 stupně. Výsledek je možné vidět na Fig. 5.

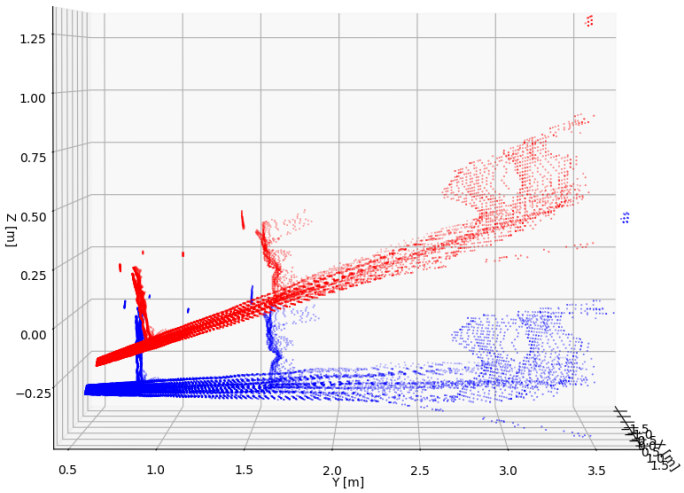

Fig. 5: Point cloud před kompenzací (červený) a po kompenzaci (modrý), zobrazen je každý 3. bod, $y <= 4$

Střed robota má dle informace o orientaci souřadných os v rovině podlahy přibližně souřadnice (0, 0).

Dále můžeme pro každý ohraničující obdelník na RGB obrázku zjištěný v sekci V zastupují nějaký objekt (překážku, bránu garáže) získat (x, y) souřadnici přiřazenou každému pixelu v něm. Tyto souřadnice jsou projekcí stěny detekovaného objektu do roviny podlahy.

Polohu každého takového objektu by však bylo vhodné reprezentovat jedním bodem. Určíme tedy medián pro obě složky souřadnic všech těchto bodů a ten prohlásíme za střed daného objektu.

Bylo by taktéž možné použít průměr, avšak experimentálně jsme ověřili, že medián má lepší výsledky, neboť ignoruje odlehlé body, jež se mohou objevit na základě nepřesností v detekci.

Prohlásit nalezený bod za střed objektu a nikoliv stěnu by mohlo u překážek, které mají válcovitý tvar, zanést chybu o velikosti jejich poloměru. Ten se však pohybuje kolem 2,3 cm, což je v rámci nejistoty vzniklé při hledání daného bodu a jeho zakreslení do mapy. Proto tuto chybu můžeme zanedbat.

Objekty, kterým jsme nepřiřadili ohraničující obdelník (garáž), nebudeme reprezentovat pouze jedním bodem, ale vezmeme všechny body prostoru nalezené pomocí jejich kontur na RGB obrázku, provedeme projekci do roviny podlahy (rovina xy) a ponecháme si pouze ty s unikátní projekcí, tzn. body se stejnou x a y složkou nahradíme jedním bodem.

Dále zodpovíme otázku z předchozí sekce V: Proč použít zmenšených ohraničujících obdelníků místo celých a jak moc je zmenšit?

Ohraničující obdelníky na RGB obrázku zpravidla obsahují kromě detekovaného objektu i jeho malé okolí, neboť jsou nejmenším obdelníkem, který ohraničuje kontury objektu, které obecně nejsou obdelníkového tvaru. Nehledě na to, že kontury samotné mohou být určeny nepřesně.

V takovém případě bychom do 3D souřadnic objektu započítávali též souřadnice z pixelů, které objektu nenáleží. V obecném případě, kdy pixelů objektu je podstatně více než pixelů okolí, je problém částečně řešen tím, že za výsledný bod bereme již zmíněný medián, který tyto odchýlené body do určité míry zanedbává. V krajních případech toto však platit nemusí, např. objekt leží částečně mimo obrázek.

Navíc jsme při testování zjistili, že point cloud a RGB obrázek nejsou přesně zarovnané, tj. ačkoliv pixel na obrázku opravdu náleží objektu, v point cloudu je mu přiřazena souřadnice okolí.

Tyto nepřesnosti můžeme obejít právě tím, že ohraničující obdelník zmenšíme. U překážek a sloupků brány se nám osvědčilo zmenšit šířku na 70 % a výšku na 90 % původní hodnoty.

Podobnou úpravu by bylo vhodné realizovat i pro garáž, avšak vzhledem k tomu, že známe pouze její obecně nepravidelnou a nekonvexní konturu, nebyli jsme toho schopni dosáhnout. Aplikujeme u ní alespoň méně efektivní postup a to takový, že ze získaných bodů vypočteme těžiště a odstraníme body, které jsou od něj dále než 0,5 metrů. Tuto hodnotu jsme určili experimentálně.

Na závěr této sekce vizte Fig. 6 pro vizualizaci point cloudu pro situaci zobrazenou na Fig. 2. Pro názornost byly body překážek obarveny na červeno a body garáže na žluto.

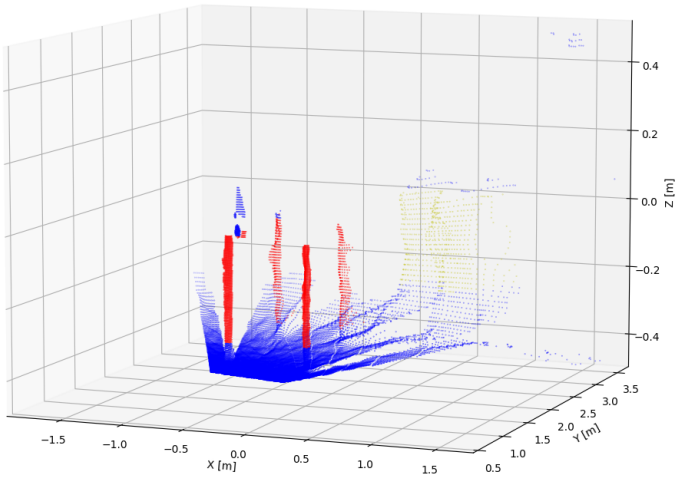

Fig. 6: Vizualizace dat z point cloudu, zvýrazněny body překážek (červená) a garáže (žlutá), zobrazen je každý 3. bod, $y <= 4$, $z <= 0,5$

VII. Mapa

Informace zjištěné během předchozích částí nyní zužitkujeme při vytváření 2D mapy (pohled shora) prostoru před robotem. Pro představu, jak bude výsledná mapa vypadat například pro situaci z Fig. 4 vizte Fig. 7.

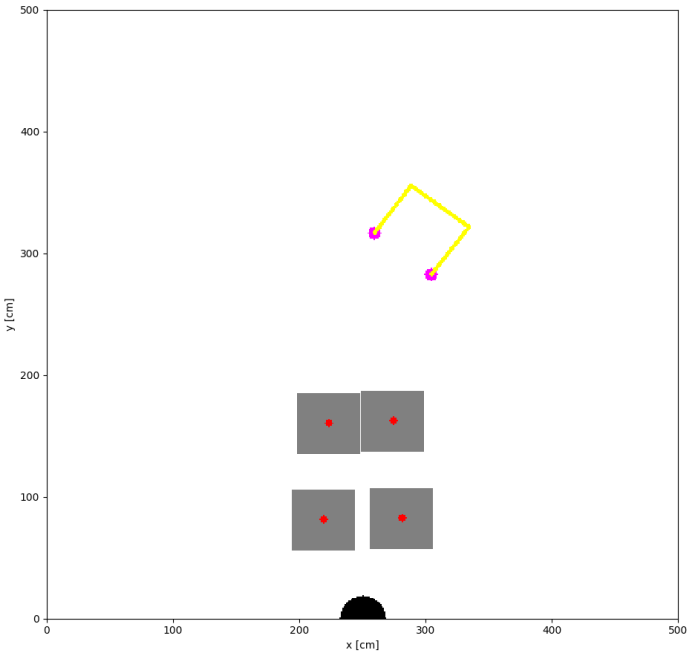

Fig. 7: Mapa s rozlišením 1 centimetr čtvereční pro prostor 5x5 metrů před robotem - robot (černá), překážky (červená), sloupky vjezdu garáže (fialová), garáž (žlutá), zakázané oblasti (šedá)

Mapu vytvoříme tak, že specifikujeme, jak velký prostor před robotem chceme zobrazovat a jaké rozlišení bude mít.

Zjistili jsme, že prostor 5x5 metrů je dostačující a není příliš velký. Výběr rozlišení znamená najít kompromis mezi přesností a rychlostí zpracování, pro náš prostor jsme experimentálně určili za vhodné rozlišení 1 centimetr čtvereční, tzn. rozlišení pro každý rozměr $r = 0,01\ m$.

Každý dílek reprezentující jeden centimetr čtvereční budeme nazývat pixelem, neboť pokud budeme chtít tuto mapu vizualizovat, bude skutečně jeden dílek odpovídat jednomu pixelu obrázku mapy.

Pro co nejjednodušší orientaci v mapě určíme souřadný systém tak, že každému sloupci pixelů přiřadíme číslo x a každému řádku přiřadíme číslo y, přičemž budou od nuly číslovány zespoda nahoru a zleva doprava. Každý pixel tedy bude odpovídat jednoznačné souřadnici (x, y).

Z informací o velikosti a rozlišení tedy vyplývá, že podél každé z os bude 500 pixelů, neboť pro převod rozměrů reálného světa do rozměrů na mapě platí:

$$\text{počet pixelů} = \frac{\text{rozměr}}{\text{rozlišení}} \tag{1}$$

Pro délku strany mapy tedy platí: počet pixelů = 5/0,01 = 500. Levý spodní pixel bude mít souřadnici (0, 0) a pravý horní pixel bude mít souřadnici (499, 499).

A. Konverze souřadnic

Před tím, než zakreslíme nalezené objekty do mapy, musíme určit souřadnice robota v ní, neboť polohu všech ostatních objektů ve světových souřadnicích známe relativně k němu, respektive ke kameře, která je připevněna téměř v jeho středu.

Abychom plně využili zorného pole kamery, umístíme střed robota do bodu (250, 0). Již dříve jsme zmínili, že robot má ve světových souřadnicích v rovině podlahy souřadnice (0, 0). Porovnáme-li tedy polohu středu robota ve světových a mapových souřadnicích, můžeme říci, že při transformaci bude v první souřadnici posunutí $t = 250$.

Na základě této informace již můžeme formulovat transformační vztah mezi světovými souřadnicemi v rovině podlahy (x_w, y_w) a mapovými souřadnicemi (x_m, y_m). Připomeneme, že rozlišení pro každý rozměr je $r = 0,01\ m$. Pak platí, že $x_m = \frac{x_w}{r} + t$ a $y_m = \frac{y_w}{r}$.

Složky (x_m, y_m) by za těchto okolností mohly nabývat i neceločíselných hodnot, proto je zaokrouhlíme. Vzniklá chyba se bude pohybovat vždy v rámci jednoho centimetru.

B. Zakreslení objektů do mapy

Po tom, co se nám podařilo převést souřadnice objektů ze světových do mapových, je můžeme zanést do mapy. Jakým způsobem to uděláme, záleží na objektu a situaci. Zároveň budeme většinou muset kromě objektu samotného zakreslit také nějaké jeho okolí, do kterého robotovi zakážeme vstup, abychom zamezili kolizi.

1) Sloupky brány garáže a překážky: Tyto objekty budeme do mapy zakreslovat jako kruh, protože jejich polohu reprezentujeme jedním bodem, jenž prohlásíme za jeho střed, a můžeme určit jejich poloměr.

Ačkoliv jsou sloupky brány jednou ze stěn garáže a nemají válcovitý tvar jako překážky, budeme je v mapě pro jednoduchost reprezentovat taktéž kruhem. Vzniklou chybu můžeme považovat za bezpečnostní rezervu, neboť kruhová reprezentace zabírá více místa, než je potřeba.

Změříme tedy potřebné rozměry těchto objektů. Jak jsme již zmínili, překážky jsou válce, za poloměr proto vezmeme poloměr jejich podstavy. U sloupků garáže, které jsou obdelníky, vezmeme za poloměr polovinu délky jejich kratší strany.

Tyto rozměry převedeme dle vzorce (1) do rozměrů na mapě.

Pro zakázané oblasti kolem překážek jsme testovali několik odlišných tvarů: kruh, různě orientovaný čtverec, hexagon, oktagon, etc. Z několika různých důvodů (např. způsob pohybu robota, spolehlivost, jednoduchost) jsme se nakonec přiklonili ke čtverci bez rotace. Jeho střed sjednotíme se středem překážky a délku jeho strany a nastavíme jako:

a = 2*(poloměr překážky + poloměr robota + bezpečnostní rezerva). Bezpečnostní rezervu určíme experimentálně.

Zakázané oblasti sloupků brány nemusíme řešit, neboť buď budou součástí zakázané oblasti celé garáže, nebo se robot nemůže dostat do situace, kdy by do nich mohl narazit. (Odůvodnění je poměrně komplikované, ve zkratce: vyplývá ze zadání úlohy a způsobu výběru cesty.)

2) Garáž: Zakreslení garáže do mapy je takřka pro všechny situace stejné: do mapy zaneseme všechny body, které jsme pro garáž zjistili, jak je popsáno v sekci VI. Odlišné jsou pouze dva případy:

- Předpokládáme, že robot není v parkovací fázi (jak je popsána v sekci III), v takové situaci by platil opět stejný způsob jako pro všechny ostatní případy.

a) **Robot vidí oba sloupky vjezdu:** Pokud nastane tato situace, nemusíme zakreslovat zjištěné body garáže do mapy a ty pak dále zpracovávat, neboť z těchto dvou bodů již můžeme vypočítat, jak bude garáž na mapě orientována. Zjistíme tedy polohu zbylých dvou rohů garáže a společně s nimi zakreslíme do mapy úsečky tam, kde by se měly nacházet stěny garáže.
Pro porovnání samotného zakreslení bodů oproti vypočteným stěnám garáže vizte Fig. 8. Vybrali jsme situaci, kdy detekce garáže byla i kvůli velké vzdálenosti garáže od robota poměrně nepřesná, abychom zvýrazili výhodu této početní metody.

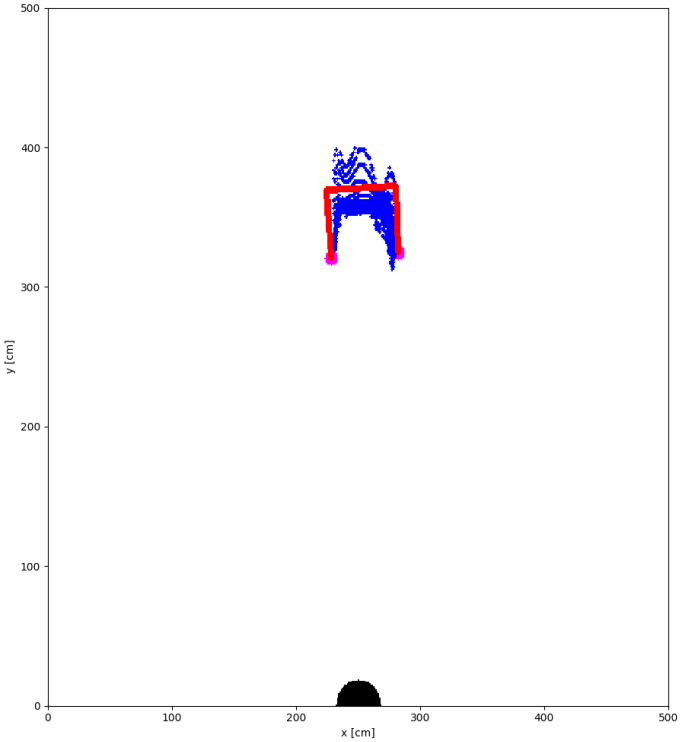

Fig. 8: Mapa - Před robotem stojí pouze garáž, původní zjištěné body garáže (modrá), vypočtené body garáže (červená)

b) **Robot nevidí ani jeden sloupek vjezdu a je dostatečně blízko garáže (jeho střed je na mapě od nejbližšího bodu garáže nejdále 65 pixelů - experimentálně zjištěná vhodná hodnota):** V takové situaci na základě proložení zjištěných bodů garáže odhadneme, kde se v prostoru přesně nachází a jakou má orientaci. Na základě toho můžeme rovnou určit polohu sloupků vjezdu do garáže.
Toto je také jediný případ, kdy kolem garáže budeme zakreslovat zakázanou oblast, neboť ji bude nutné objíždět. Ta bude mít opět tvar čtverce (tentokrát však orientovaného stejně jako garáž) se středem v centru garáže a délkou strany a takovou, že:
a = velikost delší strany garáže + 2*(poloměr robota + bezpečnostní rezerva). Bezpečnostní rezervu určíme experimentálně.

- Více o tom, jak jsme tyto dvě metody naimplementovali v sekci IX-B.

3) Robot: Již víme, že střed robota má v mapě souřadnice (250, 0). Změříme poloměr robota, ten poté převedeme do velikosti v mapě vzorcem (1) a zakreslíme kruh s daným středem a poloměrem. Na mapě bude vždy zobrazena pouze půlka tohoto kruhu, neboť robot zasahuje jednou polovinou mimo mapu.

4) Cíl: Kromě detekovaných objektů budeme do mapy zakreslovat také cíl, tzn. bod, kam má robot za úkol dojet. Cíl bude opět určet jedním bodem na mapě, pro přehlednost jej na mapu však budeme při vizualizaci zakreslovat jako kruh s poloměrem 5 pixelů. Může se stát, že se vhodný cíl určí do zakázané oblasti, nebo do jiného objektu. Pak bude nutné jeho polohu upravit. Pokud se tak stane, budeme v mapě vizualizovat oba body. Způsob určení těchto bodů naleznete v části VII-C. Pro vizualizaci vizte Fig. 9.

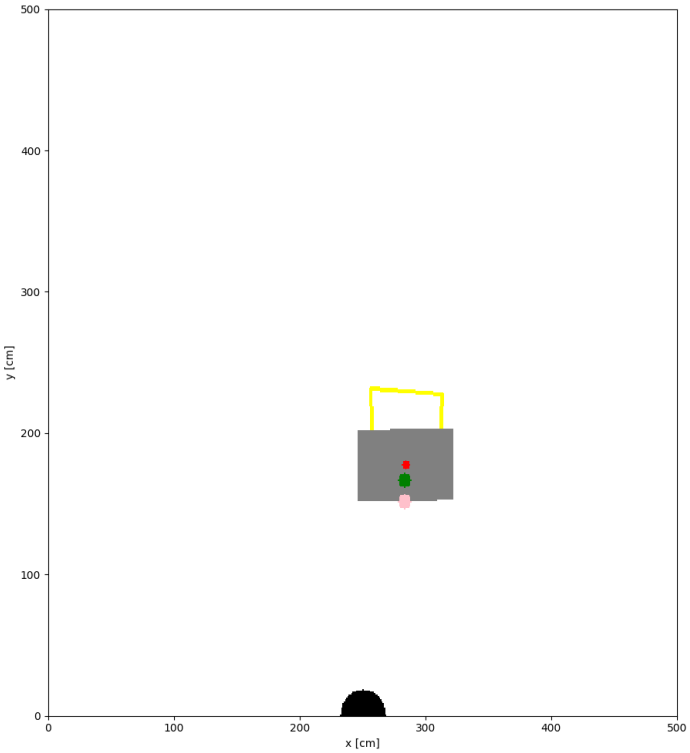

Fig. 9: Mapa - Před robotem stojí garáž, jež má před vjezdem několik překážek, původně nastavený cíl (zelená) se nachází v zakázané oblasti (šedá), přepočítaný cíl (růžová)

5) Pořadí: Dalším aspektem zakreslování objektů do mapy je pořadí, ve kterém je do mapy zakreslujeme. Zakreslované prvky překryjí ty již zanesené. Jako první do mapy zakreslíme sloupky vjezdu do garáže, dále garáž samotnou, poté překážky, robota a na závěr cíl.

Vždy nejdříve zakreslíme zakázanou oblast příslušnou danému objektu a pak teprve objekt samotný. Tento krok má pouze estetický efekt využitý při vizualizaci.

C. Určení cíle v mapě

Máme-li objekty detekované před robotem zanesené v mapě, můžeme určit bod, kam má robot dojet. Určování cíle so vztahuje pouze na případ, kdy se robot nachází ve fázi dostávání

se před vjezd do garáže. Případ, kdy je robot ve fázi parkování, je popsán v sekci IX-A - Třetí část.

Hlavním kritériem pro výběr způsobu určování cíle bude, kolik robot vidí fialových sloupků vjezdu do garáže.

- **2 sloupky:** Pokud je v mapě zanesena poloha obou sloupků vjezdu, nastavíme cíl tak, aby byl přesně mezi nimi v určité vzdálenosti od garáže. Tuto vzdálenost jsme experimentálně určili jako 25 % vzdálenosti mezi 2 sloupky vjezdu, což odpovídá přibližně 13 centimetrům. Bylo potřeba určit to tak, aby robot bezpečně nenarazil do garáže ani při příjezdu ze stran a zároveň nesrazil překážky, které můžou dle zadání stát 50 centimetrů od garáže.
- **1 sloupek:** V tomto případě určíme za referenční bod střed daného sloupku. Cíl nastavíme do vzdálenosti přibližně 40 centimentrů (experimentálně zjištěná vhodná hodnota) od referenčního bodu na přímce mezi ním a středem robota.
- **žádný sloupek:** V tomto případě budeme předpokládat, že robot vidí alespoň garáž. (Pokud nevidí ani ji, bude se točit a hledat ji.) Postup bude úplně stejný jako v případě s jedním sloupkem, pouze změníme referenční bod na nejbližší bod garáže.

- Více o tom, jak jsme tyto postupy naimplementovali v sekci IX-B.

D. Hledání cesty v mapě

Jakmile máme mapu připravenou a cíl nastavený, musíme ještě najít vhodnou cestu od robota k němu. K tomu budeme využívat obecně známého algoritmu A star, který nalezne optimální cestu v grafu.

Budeme používat euklidovskou heuristiku, avšak je možné vybrat i z několika dalších: manhattenská, Chebyshevova, oktilní, či žádná.

Počátečním bodem bude střed robota a jako koncový bod nastavíme již zmíněný cíl. Z každého bodu bude možné expandovat do všech jeho 8 okolních bodů (směry: nahoru, doprava nahoru, doprava, doprava dolu, dolu, doleva dolu, doleva a doleva nahoru), pokud tomu nezabrání jedna z omezovacích podmínek. Za omezovací podmínku bereme hranice mapy a objekty v mapě. Je možné expandovat do pixelů prázdných, či pixelů robota a cíle. Do všech ostatních je expanze zakázána.

Pokud není možné cestu do cíle určit, např. cíl je v nedostupné části mapy, robot nás na to upozorní a ukončí řešení úlohy.

Více o algoritmu A star v sekci XIII.

Nalezená cesta je množina bodů (x, y) v mapě, můžeme ji tedy čistě pro vizualizaci do mapy zanést. Výslednou mapu i s nalezenou cestou pro případ zobrazený na Fig. 2 naleznete na Fig. 10.

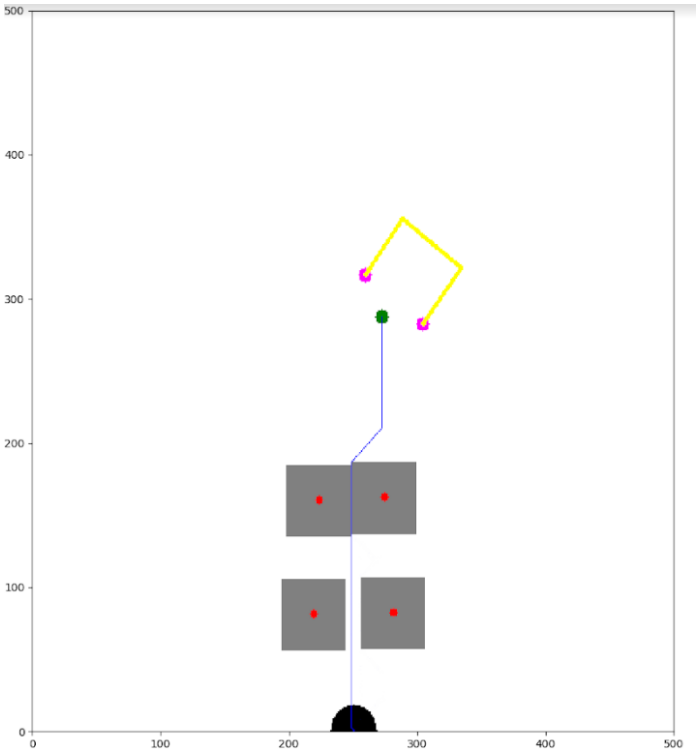

Fig. 10: Mapa se zakreslenou nalezenou cestou (modrá) k cíli (zelená)

VIII. Pohyb robota

V předchozích sekcích jsme se věnovali zpracování informací o okolí robota a jejich následnému využití k naplánování cesty. Nyní popíšeme způsob, jakým robot vykonává pohyb a tuto cestu následuje.

Pohyb robota jsme řesili dvěma způsoby. První se posléze ukázal jako nedostatečný, a tak byl nahrazen způsobem druhým.

A. Možnosti ovládání robota

Než popíšeme, jak konkrétně jsme pohyb u robota realizovali, zmíníme, jak můžeme robota ovládat a jaké informace o pohybu jsou nám k dispozici.

Společně s robotem nám bylo poskytnuto prostředí, pomocí kterého jej lze ovládat. Pro ovládání nám stačí robotovi nastavit lineární rychlost v metrech za sekundu, kterou pojede rovně před sebe, a úhlovou rychlost v radiánech za sekundu pro otáčení.

Robot nám při pohybu poskytuje tzv. odometrii. Po zapnutí, či po resetnutí odometrie, nastaví robot pro aktuální polohu souřadnici (0, 0) v souřadnicích podlahy. Souřadnice jsou udávány v centimetrech. Jak se pohybuje, poskytuje nám aktuální polohu (x, y) relativně k původně nastavenému bodu (0, 0).

Tato reprezentace se nám hodí, neboť naše mapa používá v podstatě stejných souřadnic (celočíselných). Jediný zásadní rozdíl je pouze v tom, že v odometrii je původní poloha robota (0, 0), zatímco na mapě, jak již víme ze sekce VII, je původní poloha robota (250, 0). Budeme tedy muset počítat s tímto offsetem pro souřadnici x.

B. Generace pohybových úseků z mapy

První z vyzkoušených přístupů využíval faktu, že se každý složitý pohyb v mapě dá rozčlenit na dílčí pohyby, které lze realizovat rotací a následnou translací. Tyto dílčí pohyby pak robot vykonává postupně.

U rotace i translace se kontrolovala odometrie a když dosáhla požadované hodnoty, robot konkrétní pohyb ukončil.

Hlavní problém této metody spočíval v tom, že měla potíže rozdělit složitější cesty na jednotlivé úseky ve chvíli, kdy měly být krátké, tzn. v rámci jednotek centimetrů. Jelikož se v cestě pro složitější situace (např. více překážek) začaly takové úseky vyskytovat, vymysleli jsme řešení s použitím regulátoru.

Jelikož tento způsob ve finální verzi nepoužíváme, nebudeme při jeho popisu zacházet do větších detailů.

C. Regulátor

Druhý přístup již nečlení cestu na jednotlivé úseky, bere ji jako spojitý celek. Robot ji bude následovat za pomoci regulátoru.

Chceme-li však využít regulátoru, musíme nejdříve určit veličinu, kterou budeme regulovat. Robotovi nastavíme konstantní lineární rychlost, experimentálně jsme přišli na hodnotu 0,05 m/s. Rychlost by mohla být i vyšší, ale při menší rychlosti je pohyb přesnější a spolehlivější.

K regulaci nám tedy zbývá pouze jedna ovládatelná veličina, konkrétně úhlová rychlost. Robot se bude díky nastavené konstantní lineární rychlosti pohybovat stále kupředu a bude zatáčet pomocí regulátoru, který má na starosti nastavovat právě úhlovou rychlost. Regulátor se bude snažit minimalizovat úhel mezi vektorem rychlosti robota a vektorem k dílčímu cíli. Pro představu vizte Fig. 11.

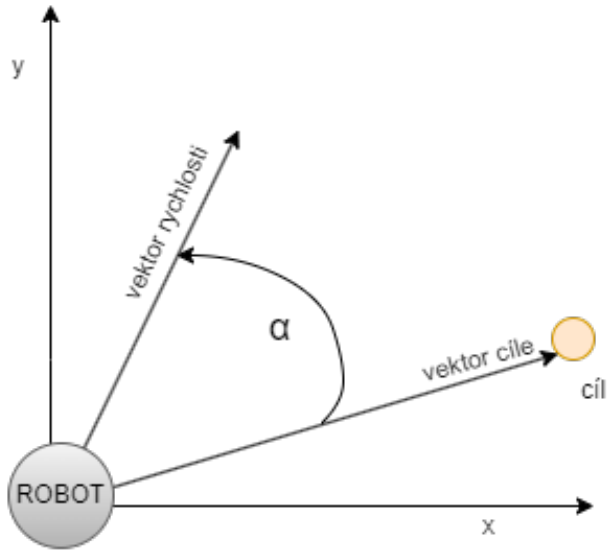

Fig. 11: Úhlová odchylka vektoru rychlosti a vektoru od robota k dílčímu cíli - chyba vstupující do regulátoru

Vektor rychlosti robota je počítaný jako diference aktuálního a předchozího bodu odometrie, tzn. odečteme od složek aktuálního bodu složky bodu předchozího.

Dále musíme určit dílčí cíl, tzn. bod, kam chceme dojet po co nejkratší cestě. Mohli bychom brát každý bod poskytnuté cesty, avšak to se ukázalo jako nepraktické, neboť jsme regulátoru měnili referenci příliš často, což vedlo k neplynulosti pohybu. Experimentálně jsme tedy určili, že z cesty budeme brát každý sedmý bod. Pokud je poslední úsek kratší než sedm bodů, vezmeme za dílčí cíl poslední bod cesty.

Dílčí cíl změníme tehdy, když se robot dostane do vzdálenosti 7 centimetrů (či menší) od akutálního dílčího cíle. Tato hodnota byla určena opět experimentálně jakožto kompromis mezi plynulostí pohybu a přesností následování cesty. U posledního bodu cesty chceme, aby robot dojel co nejblíže, proto pohyb ukončíme až tehdy, kdy je od něj robot vzdálený 1, či méně centimetrů. Pro menší vzdálenosti se stávalo, že robot bod přejel.

Vektor k dílčímu cíly tedy spočítáme jako rozdíl složek bodu dílčího cíle a složek bodu aktuální odometrie.

Regulátor samotný je typu PI. Konstanty zesilující složku I a P a další parametry byly zjištěny experimentálně. Pro I jsme zvolili hodnotu 0,05 a pro P jsme zvolili hodnotu 2. Nakonec jsme implementovali také anti-windup, který zajišťuje nulování sumační složky v případě jejího velkého nárůstu, v našem případě ve chvíli, kdy přesáhne hodnotu 20.

Tento přístup k pohybu robota se ukázal jako velmi spolehlivý pro pohyb po potřebných trajektoriích.

Pro příklad výstupu regulátoru v závislosti na vstupní chybě vizte Fig. 12.

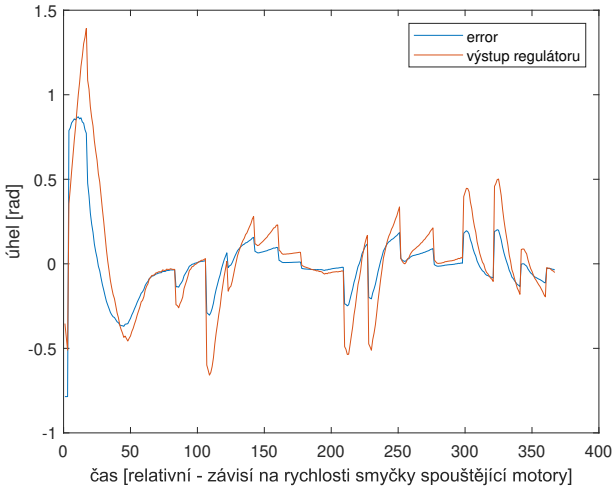

Fig. 12: Výstup regulátoru v závislosti na vstupní chybě

IX. Implementace

Výsledný program je implementován v Pythonu. Rozhraní pro ovládání robota a komunikaci s ním nám bylo poskytnuto; využívali jsme tedy již implementované funkce.

Nejdříve popíšeme strukturu programu a poté zmíníme některé způsoby implementací algoritmů a myšlenek zmíněných v předchozích sekcích.

A. Struktura programu

Program je rozčleněn do 3 základních částí, které zpracovávají jednotlivé fáze úkolu.

- **První část:** Načítáme konfigurační soubory, které obsahují veškeré konstatny použité v programu, a inicializujeme robota. Robot pak čeká na stisknutí tlačítka, aby mohl pokračovat ve vykonávání dalších fází.
- **Druhá část:** Robot se snaží dostat před vjezd do garáže. Zjednodušený vývojový diagram této části můžete vidět na Fig. 13.
  Pokud nevidí garáž, točí se po 20 stupních dokola, po každém otočení analyzuje prostředí, jak bylo popsáno v předchozích sekcích. V tom pokračuje, dokud nedetekuje garáž. Jakmile detekuje garáž, mohou nastat tři situace: (Hodnoty vzdáleností a úhlů v této části jsme určili experimentálně za vhodné.)

  – *Robot detekuje 2 fialové sloupky vjezdu:* Robot nejdříve zkontroluje, zda je již dostatečně blízko garáže. Za blízko považujeme vzdálenost menší než 65 centimetrů.
  Pokud je robot blízko garáže, pokračujeme do další části.
  Pokud není robot blízko garáže, dojede před vjezd, kde pro jistotu ověří, zda vidí alespoň jeden fialový sloupek tak, že se desetkrát otočí o 30 stupňů a analyzuje prostředí. Pokud ověří, že je před vjezdem, pokračuje do další části. Pokud nenarazil ani na jeden fialový sloupek začne druhou fázi od začátku.
  – *Robot detekuje 1 fialový sloupek vjezdu:* Pokud nastane tato situace, robot se pokusí natočit tak, aby viděl oba fialové sloupky.
  Nejdříve se otočí o 5 stupňů doprava, dále se otočí znovu o 5 stupňů doprava. Pak se vrátí na původní pozici rotací o dvakrát 5 stupňů doleva. Poté zopakuje stejný postup na druhou stranu.
  Pokud se mu to při některém z otočení podaří nalézt dva sloupky, začne druhou fázi od začátku.
  Pokud se mu to nepodaří, vidí alespoň jeden ze sloupků, protože se vrátil do původní orientace. Opět zkontroluje zda je blízko.
  Pokud je blízko, pokračuje do další části, neboť víme, že je blízko před vjezdem.
  Pokud není blízko, dojede ke sloupku, který vidí. Opět ověří, zda vidí alespoň jeden fialový sloupek, a poté buď pokračuje do další části, nebo začíná druhou část znovu.
  – *Robot nedetekuje žádný fialový sloupek vjezdu:* V této situaci víme, že je detekovaná alespoň garáž.
  Robot nalezne takovou orientaci, aby viděl co nejvíce žlutých bodů garáže. Začne se otáčet přibližně o 2 stupně doprava tak dlouho, dokud se mu garáž neztratí ze zorného pole (nevidí ani jeden žlutý bod garáže). V takové situaci si můžeme být jistí, že garáž je celá nalevo od robota.
  Následně se tedy začne otáčet doleva opět po 2 stupních a po každém otočení spočítá počet bodů garáže. Robot se bude otáčet doleva do té doby, dokud počet nalezených bodů bude stoupat. Jakmile se stane, že bude počet menší než při předchozím snímku, otočí se robot naposledy doprava, aby se vrátil do pozice, kdy viděl nejvíce bodů garáže. Až dokončí tuto operaci, ověří, zda je blízko garáže.
  Pokud je blízko, objede garáž tak, aby se dostal před její vjezd. Poté opět ověří, zda je před vjezdem, přičemž buď pokračuje do další části, nebo začíná druhou část od začátku.
  Pokud není blízko, dojede k nejbližšímu bodu garáže a začíná druhou část od začátku.

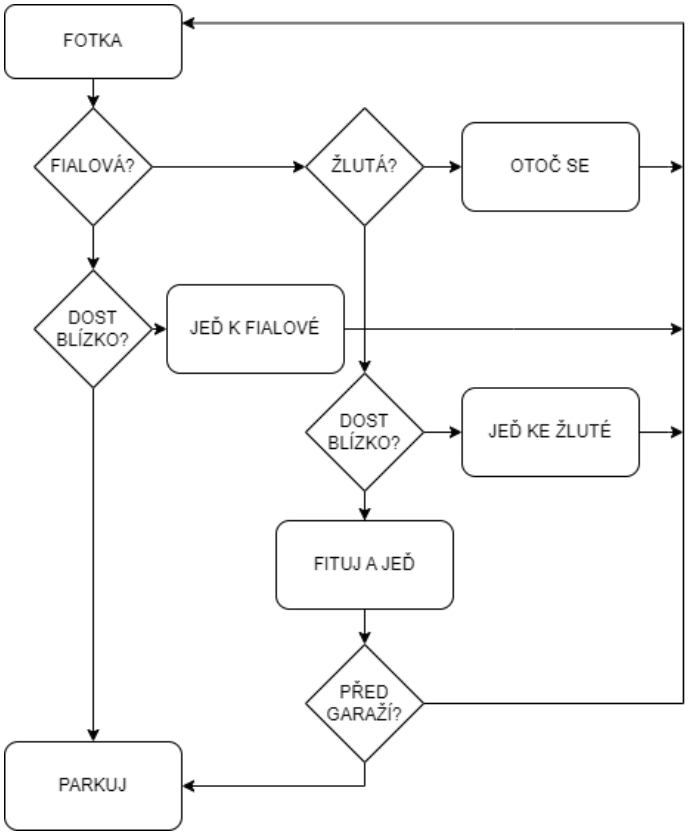

Fig. 13: Vývojový diagram druhé části programu: při odpovědi ANO se posouváme v diagramu dolu, při odpovědi NE se posouváme v diagramu doprava

- **Třetí část:** V této fázi se robot snaží zaparkovat do garáže. Pro jednodušší popis se budeme odkazovat k ilustraci na Fig. 14. Na ní je možné vidět robota (šedá), důležité body cesty (červená, oranžová a zelená), bod pro referenci (modrá), sloupky vjezdu (fialová) a garáž (žlutá).
  Robot se již nachází v okolí vjezdu tak, že je možné aby viděl (ne nutně najednou) dva fialové sloupky. Může stát například na místě jako na ilustraci. Robot se začne po přibližně 2 stupních otáčet doprava, při každém otočení pořídí fotku a hledá na ní žluté body garáže, které zjistíme způsobem popsaným v sekci VI. Jakmile garáž najde, zopakuje postup, který jsme využili ve druhé fázi pro hledání orientace robota vůči garáži takové, aby robot viděl co největší část garáže, což se nám bude hodit pro další postup.
  Dále získáme všechny body garáže na mapě způsobem popsaným v předchozích sekcích. Těmito body se

pokusíme proložit dvě přímky. Pokud se nám to nepodaří, robot nás na to upozorní, otočí se o náhodný úhel, abychom začínali z jiné pozice a byla tak větší šance, že nový pokus dopadne jinak, a zopakuje celou operaci od hledání maxima žlutých bodů garáže.

Pokud se podaří proložit body garáže dvě přímky, zkontrolujeme, zda jsou na sebe kolmé tím, že změříme úhel mezi nimi. Experimentálně jsme zjistili, že můžeme nastavit poměrně velkou toleranci (15 stupňů) pro odchylku od devadesáti stupňů, které by mezi sebou měly mít. Pokud zjištěný úhel nebude v toleranci, robot bude pokračovat stejně jako v případě, kdy nemohl proložit nalezenými body dvě přímky.

Pokud úhel v toleranci je, nalezneme jejich průsečík a ten prohlásíme za jeden z rohů garáže. (modrý referenční bod na ilustraci) Dále na základě faktu, že zadní stěna garáže bude vždy více vodorovná, tzn. úhel mezi ní a osou x bude menší než u druhé, určíme, která přímka reprezentuje zadní stěnu a která zas stranu postranní. Poté změříme skutečné rozměry brány a ty pak převedeme dle vzorce (1) do mapových rozměrů.

Na základě těchto informací můžeme nalézt v mapě prostředek obou stěn. Od referenčního bodu (roh garáže) se posuneme vždy o polovinu zjištěné délky dané stěny ve směru, který udává proložená přímka. Díky tomu nalezneme střed garáže (zelený bod na ilustraci). Do středu garáže budeme chtít robota dostat.

Abychom zabránili kolizi mezi robotem a garáží při parkování, nalezneme bod uprostřed před bránou. (červený bod na ilustraci) Nalezneme jej tak, že definujeme přímku procházející zeleným bodem ve směru postranní stěny garáže, na ní pak určíme nejbližší bod ke středu robota.

Na hledání cesty použijeme algoritmus A star zmíněný již v sekci VII. Vzhledem k tomu, jakým způsobem hledá tento algoritmus cestu mezi dvěma body v mapě s osmiokolím (lze zatáčet pouze v násobcích 45 stupňů), interpolujeme rovnoměrně úsečku mezi bodem před brankou (červený) a bodem ve středu garáže (zelený) pěti body (oranžové + zelený a červený), abychom zajistili, že se robot nevychýlí a nenarazí do garáže.

Mezi po sobě jdoucími body počínaje středem robota poté nalezneme cestu pomocí A staru a tyto jednotlivé úseky spojíme do jedné cesty, kterou bude robot následovat. Dostane se po ní do středu garáže, kde vydá zvukový signál, čímž úspěšně zakončí zadaný úkol.

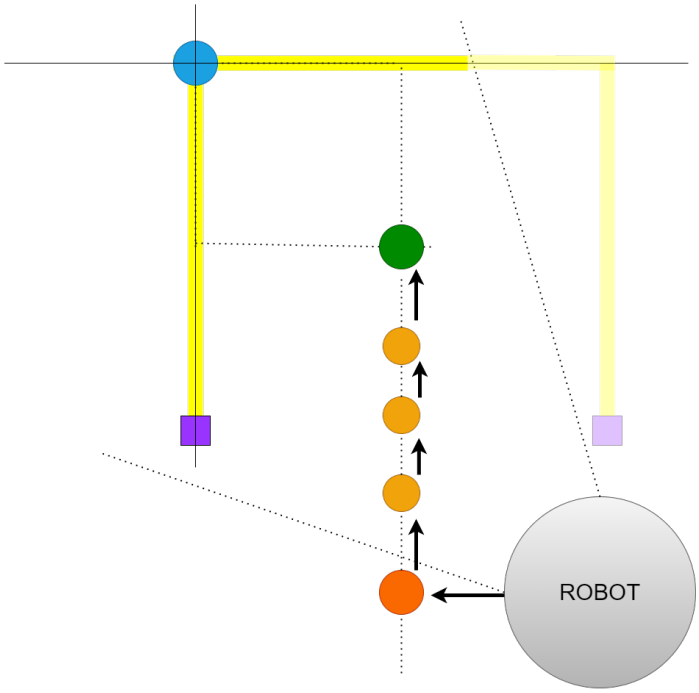

Fig. 14: Nákres postupu při parkovací sekvenci

Pozn. Pokud dojde ke zmáčknutí předního nárazníku během některé z těchto fází, robot se okamžitě zastaví.

B. Algoritmy z předchozích částí

V této části popíšeme některé výpočetní algoritmy, které jsme používali v předchozích sekcích. Název následujících podsekcí bude zvolen tak, aby odpovídal problému, u kterého jsme se na toto dovysvětlení odkazovali.

1) Detekce objektů na RGB obrázku: Funkce, které jsme pro zpracování RGB obrázku používali, nalezneme v knihovně *OpenCV*. Pro blížší podrobnosti k těmto funkcím viz sekce Přílohy.

Na konvertování obrázku z RGB do HSV formátu jsme používali funkci *cvtColor()*.

Pro hledání kontur jsme využili funkce *findContours()*.

Pro určování ohraničujících obdelníků jsme využili funkce *minAreaRect()*.

Toto jsou hlavní z používaných funkcí. Pro další vizte samotnou implementaci.

2) Výpočet orientace a pozice garáže ze znalosti pozice 2 sloupků vjezdu: Nejdříve změříme rozměry garáže a ty pak převedeme do rozměrů v mapě pomocí vzorce (1).

Dále vypočteme orientaci brány v mapě. Sloupky jsou reprezentovány každý jedním bodem: x_1 bude sloupek víc vlevo (menší x souřadnice) a x_2 bude sloupke víc vpravo (větší x souřadnice). Získáme vektor mezi nimi $v = x_1 - x_2$. A na základě toho můžeme získat úhel α, který svírá kladná část osy x a přímka procházející oběma body, pomocí funkce *atan2()*.

Vezmeme bod x_1 a vypočteme z něj za pomoci trigonometrie, či transformací posun o úhel α a zjištěnou délku přední strany garáže. Tak se dostaneme do bodu x_2. Další bod nalezeneme tak, že se stejným způsobem posuneme z bodu x_2, tentokrát však o úhel α + 45 stupňů a délku boční strany garáže. Stejným způsobem pak z bodu x_3 nalezneme bod x_4.

3) Proložení bodů garáže obdelníkem: Při pohledu zezadu na garáž můžeme vidět vždy maximálně 2 strany najednou.

Nejprve vezmeme body garáže a pokusíme se nalézt jednotlivé strany. Na zjištěné body použijeme dvakrát metodu RANSAC. Pro oba průběhy použijeme stejné parametry: 300 iterací, inliers počítáme v rámci 5 centimetrů od fitnuté přímky a za minimální počet inliers pro validní fit bereme 20 bodů.

Pro každou nalezenou přímku spočítáme krajní body. Vezmeme 2 body z jejich inlierů (jeden nejvíc vpravo a druhý nejvíc vlevo) a dosadíme je do předpisu dané přímky.

Mohou nastat dvě možnosti:

a) Nalezneme 2 přímky: Spočteme polohu průsečíku těchto dvou přímek. Na základě toho spočteme délky fitnutých stran jako vzdálenost průsečíku od vzdálenějších krajních bodů. Ty pak porovnáme se změřenými délkami stran garáže převedenými do rozměrů v mapě. V obecném případě se nebudou nikdy rovnat.
Budeme tedy muset rozhodnout, která z fitnutých stran odpovídá které straně garáže. Je možné využít několik různých kritérií. My se rozhodujeme na základě rozdílů fitnutých délek od skutečných.
Jakmile rozhodneme, můžeme od průsečíku fitnutých přímek, který bereme za jeden z rohů garáže, nalézt na těchto přímkách ve vzdálenostech, pro které jsme se rozhodli v předchozím kroku, další dva rohy garáže. Poslední pak dopočteme stejným způsobem jako v předchozí podsekci IX-B3.

b) Nalezneme 1 přímku: Spočteme vzdálenost krajních bodů od sebe. Na základě toho se rozhodneme, zda je před robotem kratší, či delší strana. (Které bude spočtená délka blíže.) Máme tedy dva body a potřebné délky. Pro dopočítání zbylých dvou použijeme stejný postup jako v podsekci IX-B3.

Bližší informace o RANSAC algoritmu nalezenete v sekci XIII.

X. Diskuse

Během řešení jsme narazili na určitá omezení, která vyplynula až při řešení daného problému. Krátce je nyní zmíníme.

Vzhledem k tomu, že se při každé nové analýze prostoru kolem robota přepíše stará mapa, nemáme možnost brát v potaz objekty mimo zorné pole kamery. Proto je potřeba aby překážky byly od sebe alespoň 50 centimetrů všude a ne pouze 1 metr od garáže, jak bylo napsáno v zadání. To však neznamená, že pokud nebude toto kritérium splněno, robot nutně nedokáže zadání splnit.

Dále jsme chtěli implementovat zastavení na nečekané překážky, které bychom detekovali pomocí point cloudu: robot by zastavil, pokud by se objevil objekt v definované nebezpečné vzdálenosti. Bohužel se nám to nepodařilo, neboť nemáme zkušenosti s ROS frameworkem a když jsme se to pokusili naimplementovat, focení point cloudu vytvářelo taková zpoždění pro reakci koleček, že regulátor začal jinak plynulý pohyb řídit velmi trhaně a nepřesně.

Tyto nedostatky by bylo možné vyřešit. Bohužel nezbyl čas na jejich implementaci.

Se záměrem stručnosti neobsahuje tato zpráva veškeré postupy, které jsme při řešení zadaného problému vyzkoušeli. Popisuje pouze finální implementovanou verzi.

XI. Závěr

Podařilo se nám splnit zadání, jak jsme si ho definovali v sekci III. Uvědomujeme si však, že by řešení mohlo být robustnější a že je na naší implementaci mnohé co zlepšovat.

XII. Doporučení do dalších let

Z úlohy máme celkově pozitivní pocit, neboť jsme se toho mnoho naučili.

Myslíme si, že by studentům mohlo usnadnit práci, pokud by jim bylo dovoleno použít LIDAR, který na robotovi již je.

XIII. Přílohy

Odkazy s podtržítkama v názvu je potřeba zkopírovat manuálně, neboť LaTex na ně neumí správně odkazovat.

- BLižší specifikace zadání:
  https://cw.fel.cvut.cz/wiki/courses/b3b33lar/tasks/garaz_cs
- Bližší specifikace robota:
  https://gitlab.fel.cvut.cz/robolab/lar/-/tree/master
- Další informace o A star algoritmu:
  https://cs.wikipedia.org/wiki/A*
- Další informace o RANSAC algoritmu:
  https://www.baeldung.com/cs/ransac
- OpenCV dokumentace:
  https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html
- Odkaz na GitHub:
  https://github.com/kocajan/TURTLE_FRANKLIN