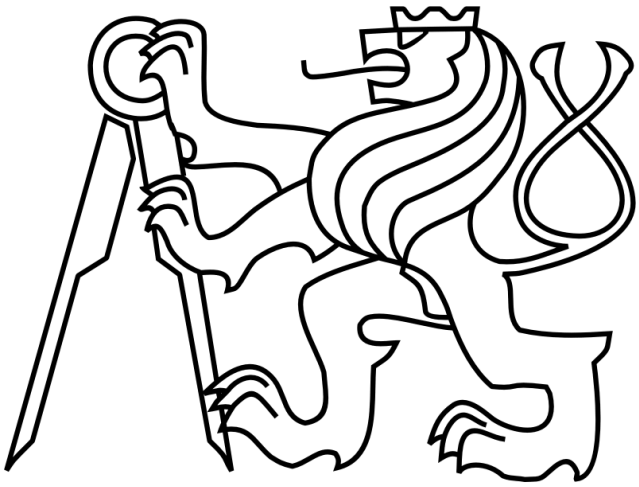
České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra kybernetiky

BAKALÁŘSKÁ PRÁCE



Mapování vnitřního prostředí autonomní helikoptérou

Autor: David Česenek

Vedoucí práce: Ing. Jan Chudoba

Praha, červen 2016

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne ....

Poděkování

Na tomto místě bych chtěl poděkovat především vedoucímu této práce, Ing. Janu Chudobovi, za přátelský přístup, podnětné rady a připomínky, ochotu a často i velkou trpělivost.

Za podporu i povzbuzení při celém studiu děkuji především svým rodičům, celé své rodině a přátelům.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: David Č e s e n e k

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Mapování vnitřního prostředí autonomní helikoptérou

Pokyny pro vypracování:

Předmětem bakalářské práce je návrh a implementace metody mapování prostředí helikoptérou.

Seznamte se s aktuálně používanými metodami mapování vnitřních prostředí pro pozemní, nebo létající roboty. Zaměřte se na metody stavby mapy, která bude později použita pro navigaci helikoptéry. Po konzultaci s vedoucím práce definujte nutné omezující podmínky s ohledem na řešitelnost práce.

Vyberte, nebo navrhněte vhodnou metodu pro implementaci na helikoptéře poskytnutou pracovištěm vedoucího. Instalujte na helikoptéru potřebné senzory, implementujte navrženou metodu a proveďte experimentální ověření a vyhodnocení její funkce.

Seznam odborné literatury:

[1] Mázl Roman: Lokalizace pro autonomní systémy. Disertační práce, Praha, 2007.
[2] Tomáš Báča: Řízení vzájemně lokalizovaných bezpilotních helikoptér. Diplomová práce, Praha, 2013.

Vedoucí bakalářské práce: Ing. Jan Chudoba

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 17. 12. 2015

Název práce: Mapování vnitřního prostředí autonomní helikoptérou

Autor: David Česenek

Katedra (ústav): Katedra kybernetiky

Vedoucí bakalářské práce: Ing. Jan Chudoba

e-mail vedoucího: jan.chudoba@ciirc.cvut.cz

Abstrakt: V této práci představujeme řešení pro mapování vnitřního prostředí UAV. Pro implementaci byla použita upravená kvadrokoptéra doplněná o laserový dálkoměr, interciální měřící jednotku (IMU), modul Px4flow s kamerou měřící optický tok a kompaktní počítač. Vytvořený program v reálném čase přímo na počítači na kvadrokoptéře sestavuje 2D mapu prostředí a určuje aktuální pozici. 2D mapování je založené na ICP-SLAM algoritmu, který využívá jaho hlavní senzor laserový dálkoměr. Měření z laserového scanu jsou kompenzována o náklony kvadrokoptéry podle dat z IMU. Jako náhrada klasické odometrie byl implementován odhad pozice z optického toku kamery. Provedené experimenty prokázaly funkčnost vytvořeného programu v několika typech vnitřního prostředí. Vytvořený program lze použít jako součást řešení pro budoucí plně autonomní činnost kvadrokoptéry.

Klíčová slova: mapování vnitřního prostředí, simultální lokalizace a mapování (SLAM), unmanned aerial vehicle (UAV), ICP

Title: Indoor mapping by autonomous helikopter

Author: David Česenek

Department: Department of Cybernetics

Supervisor: Ing. Jan Chudoba

Supervisor's e-mail address: jan.chudoba@ciirc.cvut.cz

Abstract In this work we present indoor-mapping solution for an quadrotor unmanned aerial vehicle (UAV). The custom-built quadrotor UAV equipped with laser rangefinder, inertial measurement unit (IMU), Px4flow module with optical flow camera and onboard computer was used. Our software creates realtime 2D maps of indoor environment and estimates UAV's position in map. It is based on ICP-SLAM algorithm using laser rangefinder as a main sensor. Laser measurements are enhanced by IMU data. Optical flow camera is used for gross position estimation as a substitution for odometry. Various indoor experiments have confirmed that our solution works and it could be used in the future as a part of the solution for autonomous operation tasks.

Keywords: indoor-mapping, simultaneous localization and mapping (SLAM), unmanned aerial vehicle (UAV), ICP

Obsah

1 Kapitola 1.
Úvod

Poslední desetiletí lze bezesporu označit jako období obrovského pokroku na poli mobilní robotiky a výpočetní techniky vůbec. Strmý nárůst výpočetního výkonu, stálé zlepšování zdrojů energie i používaných senzorů spolu s celkovou miniaturizací a klesající cenou způsobují masivní rozšíření mobilní robotiky. I v běžném životě se tak začínáme setkávat s kdysi drahými a nedostupnými technologiemi.

Záležitostí posledních několika let jsou také bezpilotní helikoptéry, nebo také kvadrokoptéry, multikoptéry, drony či obecněji UAV[1], MAV[2]. Jeden z představitelů vyšší třídy běžně prodávaných kvadrokoptér je uveden na obrázku 1.1.

Terminologie v této oblasti není jednotná a běžně se uváděné pojmy zaměňují. Za "*autonomní helikoptéru*" budeme v této práci považovat UAV které lze popsat následovně.

Z mechanického hlediska zařízení disponuje většinou čtyřmi či šesti vrtulemi, které bývají osazovány symetricky na samostaných ramenech pevné platformy. Jako pohony vrtulí se používají malé, ale výkonné BLDC elektromotory[3], u kterých lze externími regulátory snadno řídit otáčky. Srdce celého systému pak tvoří řídicí jednotka, která na základě informací ze senzorů a pokynů přijímaných bezdrátově od pilota reguluje otáčky všech motorů, tak, aby se celá platforma pohybovala požadovaným způsobem. Vzhledem k těmto charakteristickým rysům mají tato UAV kolmý způsob vzletu i přistávání.

Autonomní helikoptéra by měla disponovat schopností určit svoji polohu, určit si cíl svého letu (např. požadavek kompletního zmapovanání nějaké oblasti), a nakonec se do tohoto cíle dostat (např. samostatně realizovat mapování neznámého prostředí). Budeme tedy předpokládat, že popisovaná helikoptéra implementuje navíc také programovatelný počítač, který je schopen

[1] Z anglického Unmanned Aerial Vehicle, označuje obecně bezpilotní letadlo, které může být řízeno na dálku, nebo létat autonomě.

[2] Z anglického Micro Air Vehicle, obecné označení pro UAV malých rozměrů.

[3] BLDC motor (Brushless DC motor) je stejnosměrný motor, u kterého je speciálním obvodem zajišťována komunutace (samotný točivý stroj tedy pracuje v důsledku se střídavým proudem).

získávat měřená data ze senzorických systémů, softwarově je zpracovávat, komunikovat s řídicí jednotkou a v důsledku tak ovlivňovat řízení celého systému.

V této práci budeme výše popsanou autonomní helikoptéru nadále označovat zkráceně jako *kvadrokoptéru* či *helikoptéru*. Pojem UAV bude používán v obecnějším slova smyslu pro popis libovolného bezpilotního létajícího prostředku.

Obrázek 1.1.: DJI Phantom 3 Proffesional, převzato z `http://www.dji.com`

Především díky svým řádově nižším nákladům a podstatně menší velikosti ve srovnání s klasickými letady nalézají UAV uplatnění v řadě různých oborů a aplikací. A to především tam, kde by jiné způsoby byly příliš drahé či riskantní.

Velmi populární je používání pro záznam videí. U dražších výrobků se též setkáváme s přímým přenosem snímaného videa k pilotovi. Tento způsob filmování byl již mnohokrát úspěšně použit i u řady známých filmů z posledních let [5]. Monitorování UAV se prosazuje také v průmyslu, kde slouží například pro dohled nad staveništěm.

Opravdu netradiční využití kvadrokoptér pak publikovali autoři z University ETH v Zurichu [6]. Se skupinou malých upravených kvadrokoptér realizovali v laboratorních podmínkách stavbu několik metrů dlouhého lanového mostu. Most byl po úplném spletení funkční a unesl i dospělého člověka. Ačkoliv autoři pokus realizovali za použití velmi přesného externího kamerového lokalizační systému, jedná se o projekt, který dokazuje pestrost možného využití UAV.

Velká univerzálnost a skvělá mobilita předurčují UAV i pro použití při *mapování* prostředí. Na rozdíl od prostého záznamu videa při letu je mapování o poznání kompexnější disciplínou, která vyžaduje řešení celé škály podproblémů z různých oblastí. Počínaje nutností lokalizace kvadrokoptéry při letu, přes vyhodnocování dat ze senzorů v reálném čase až po strojové vytváření samotné mapy ze získávaných dat.

Mapování se v této práci budeme věnovat ze dvou různých úhlů pohledu:

- Z pohledu kartografie budeme pod tímto pojmem spatřovat snahu prozkoumat nám neznámé prostředí. V důsledku to tedy znamená, získat takovou mapu, která je na první pohled pochopitelná člověku.
- Mapováním v kontextu mobilní robotiky budeme rozumět především snahu samotného robotu sestavit si takový model okolního prostředí, aby v něm byl schopen lokalizovat svoji pozici a plánovat svoji činnost.

Oba přístupy spolu souvisejí a důraz na jejich vzájemné propojení bude kladen i v této práci. Na základě mapování v kontextu mobilní robotiky tak budeme požadovat i člověku srozumitelný výstup.

Struktura této práce je následující.

Nejprve v kapitole 2 stručně definujeme základní pojmy jako *mapování* a *lokalizace*, na základě kterých formálně zavádíme pojem „Simultální lokalizace a mapování“. Kapitola mimo jiné rozebírá základní způsoby modelování vnitřního prostředí.

Kapitola 3 je věnována aktuálnímu přehledu použitelných senzorických systémů pro realizaci mapování. Jsou zde popsány základní fyzikální principy senzorů i z toho plynoucí omezení při použití. Je kladen důraz na vzájemné porovnání kladů a záporů pro plánované použití na UAV.

Následující kapitola 4 rozebírá konkrétní způsoby řešení problému Simultální lokalizace a mapování. Závěr této kapitoly je věnován přehledu současného stavu problematiky mapování vnitřních prostředí UAV ve světě.

Jádrem této práce je kapitola 5, která představuje námi navržené řešení mapování. Neprve krátce popíšeme harwarové vybavení realizované kvadrokoptéry. Definujeme omezující podmínky pro řešitelnost této práce a diskutujeme výběr vhodné metody mapování. Dále navrhujeme vylepšení vybrané metody pro účely implementace na kvadrokoptéru a popisujeme softwarovou implementaci použitých knihoven i vlastních zdrojových kódů.

Implementovaná metoda byla testována v řadě experimetů, jejichž výsledky shrnuje a hodnotí kapitola 6.

V závěru této práce, kapitole 7, je celá tato práce zhodnocena a uvádíme také nápady na další možné rozšíření.

Kapitola 2.

2 Mapování prostředí

2.1. Lokalizace a mapování

Mapováním rozumíme sestavování modelu okolního prostředí - *mapy*. Z pohledu robotu můžeme na celý proces nahlížet jako na snahu uložit abstraktní model okolního světa do nějaké vhodné datové struktury v paměti. Mapa prostředí pak může představovat referenci, vůdči které mobilní robot určuje svoji pozici, případně může být požadována přímo jako jeden z výstupů činnosti robotu (požadujeme lidem srozumitelnou mapu). Podrobněji jsou mapy a způsoby jejich budování rozebrány v následující podkapitole 2.2.

Lokalizací rozumíme určení souřadnic robotu (jeho polohy) v daném prostředí. Pokud se robot pohybuje pouze v jedné rovině, jedná se zpravidla o určení dvou kartézských souřadnic $[x, y]$ a úhlu natočení robotu θ. Pokud budeme uvažovat pohyb v třírozměrném prostoru (např. obecný pohyb UAV), jsou zpravidla odhadovány tři kartézské souřadnice $[x, y, z]$ případně i další tři úhly[1], určující orientaci robotu na daném místě (těleso v 3D prostoru má 6 stupňů volnosti). Poloha robotu je vždy vztahována k nějaké referenci. Zpravidla se jedná o mapu prostředí, případně pouze o počáteční polohu robotu. V literatuře se dále úlohy lokalizace rozdělují na tzv. sledování polohy (jsou podstatné pouze relativní souřadnice robotu vůdči nějaké počáteční poloze, mapa prostředí není nezbytně nutná) a tzv. globální lokalizaci (robot určuje svoje absolutní souřadnice v modelu okolního prostředí, který je předem známý) [31].

Mapování i lokalizace vycházejí z informací, získaných o prostředí vhodnými senzorickými systémy. Jsou to buď senzorické systémy, poskytující informaci o vzdálenosti robotu vzhledem k okolnímu prostředí, nebo senzorické systémy měřící pohyb (polohu) robotu jako takového (viz kapitola 3).

Mapování patří spolu s lokalizací robotu v daném prostředí mezi základní dovednosti, kterými musí autonomní mobilní robot disponovat. Jak uvádí autor ve své práci [31], mapování i loka-

[1] Tyto úhly bývají zpravidla označovány jako „yaw, pitch, roll“ - rotace kolem os souřadného systému z, y, x.

lizace jsou pro autonomní mobilní robot naprosto klíčové, má-li robot v daném prostředí vykonávat nějakou smysluplnou činnost. Mapování v kontextu mobilní robotiky tak není pouze záležitostí specializovaných robotů pro průzkum neznámého prostředí, ale komplexním problémem, se kterým se musí potýkat řada autonomních mobilních robotů.

Mapování spolu s lokalizací jsou do značné míry vzájemně komplementární. Je to dáno tím, že přesnost mapy a lokalizace spolu úzce souvisejí. Přesná lokalizace je nezbytným předpokladem pro přesné mapování a naopak.

Pro průzkum a mapování neznámého prostředí autonomním mobilním robotem jsou současná lokalizace a mapování klíčové. Tímto problémem a jeho řešením se zabývají techniky označované souhrně jako tzv. *Simultální lokalizace a mapování*, které jsou definovány v podkapitole 2.3.

2.2. Typy map

Mapa v kontextu mobilní robotiky je budována na základě senzorických měření. Na takto vytvářenou mapu klademe tři základní požadavky [39]:

- Mapa by měla být dostatečně efektivně uložena, tak, aby ji bylo možné snadno použít, např. pro lokalizaci, plánování ap.
- Mapa by měla reflektovat předpokládané použití, stejně jako použité senzorické systémy a charakter mapovaného prostředí.
- Mapa by měla nějakým způsobem uchovávat informace o nejistotách.

Jak vyplývá z druhého bodu, neexistuje univerzální mapa. Kvůli rozdílným podmínkám mapování i rozdílným požadavkům na míru abstrakce je nezbytně nutné používat různé přístupy při tvorbě map. V následujícím přehledu se zaměříme pouze na mapy, použitelné ve vnitřním prostředí a kromě samotného popisu map vždy stručně nastíníme způsob, jakým jsou tyto mapy vytvářeny.

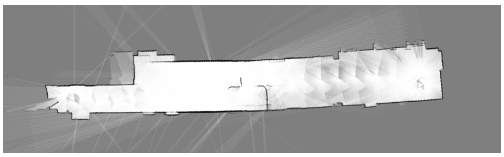
Mezi patrně nejpoužívanější mapy zde patří tyto základní typy [31]:

- mřížky obsazenosti
- geometrické mapy
- příznakové mapy

Obecně lze říci, že s rostoucí abstrakcí roste odstup od prostých senzorických měření a zpravidla také náročnost strojového vytváření této mapy. Na druhou stranu ale klesají nároky na paměť nutnou pro uložení mapy. Ilustrace podoby některých map jsou na obrázku 2.1.

2.2.1. Mřížky obsazenosti

Mřížky obsazenosti patří mezi mapy s pravděpodobnostním přístupem a s nízkou mírou abstrakce. Někdy bývají tyto mapy označovány obecněji také jako *senzorické mapy*, protože se pro



(a) Mřížka obsazenosti exportovaná do grafické podoby.

(b) Geometrická mapa, převzato z [15]

Obrázek 2.1.: Některé typy map

jejich vytváření používají přímo surová, případně lehce upravená, senzorická data. Celý prostor je rozdělen ekvidistantně na tzv. buňky, typicky malé čtvercové oblasti (pokud uvažujeme 3D variantu mapy tak malé krychle). Každá buňka má kromě své pevně definované polohy také pravděpodobnost, zda se na její pozici nachází překážka či volný prostor. Počet buněk na jednotku plochy (či prostoru) určuje rozlišení mřížky obsazenosti.

Budování mřížky obsazenosti je zpravidla realizováno iterativně ve dvou krocích [23]. Nejprve je stanoven pravděpodobnostní model použitého senzoru, který určuje rozložení pravděpodobnosti jednotlivých buněk mřížky. Ve druhém kroku je na základě modelu senzoru a aktuálního senzorického měření aktualizována mřížka obsazenosti, většinou za použití Bayesovy věty o úplné pravděpodobnosti. Zpravidla se uvažuje, že pravděpodobnosti obsazení jednotlivých buněk mřížky jsou nezávislé.

Velkou výhodou mřížek obsazenosti je jejich univerzálnost a jednoduchost. Snadno z nich lze získat nejen grafickou reprezentaci srozumitelnou člověku, ale lze je také převést na geometrickou mapu. Díky pravděpodobnostnímu přístupu lze do mapy zahrnout nejistoty senzorických měření. Jak uvádí ve své práci práci autor [31], mřížky obsazenosti mohou dokonce sloužit jako nástroj pro fúzi dat z různých senzorických systémů. Na druhou stranu, kvadratický růst paměťové náročnosti s velikostí mapovaného prostoru nedovoluje použití mřížek obsazenosti pro rozsáhlá prostředí.

2.2.2. Geometrické mapy

Geometrické mapy nabízejí vyšší míru abstrakce než mřížky obsazenosti. Z hrubých senzorických dat či mřížek obsazenosti jsou extrahovány hrany (úsečky), ze kterých je poté sestavován model prostředí. Existují i tzv. *polygonální mapy*, které pracují obecněji přímo s geometrickými primitivy, které pak smysluplněji reprezentují objekty (překážky) v prostoru. Pro vytvoření geometrické mapy je tedy klíčový algoritmus hledání hran v senzorických datech.

Jako jeden z možných algoritmů zmiňují autoři [33] RANSAC[2] algoritmus, který z určité ob-

[2] Z anglického „Random Sampling Consensus“.

lasti senzoriského měření vezme náhodně nějaký vzorek dat a optimalizační metodou nejmenších čtverců hledá co nejlepší aproximaci přímky. Pro každou aproximaci je registrován počet odpovídajících bodů v měření a počítána ztrátová funkce. Algoritmus končí po určitém počtu iterací, jako úsečkový segment je poté označen segment z iterace s nejvíce korospondujícími body, pokud bylo dosaženo definovaných postačujících podmínek pro nalezení úsečkového segmentu.

Dalším algoritmem pro nalezení úsečkových segmentů je algoritmus „Split and Merge“, který také používá metodu nejmenších čtverců pro nalezení aproximace přímky, ale aproximace je prováděna rekurzivně nejprve na celém senzorickém měření a poté na jeho stále menších částech [39].

Geometrické mapy jsou snadno srozumitelné člověku a vyžadují podstatně méně paměti než mřížky obsazenosti. Je tedy možné jejich použití i pro rozsáhlejší prostředí. Navíc mohou vykazovat vyšší přesnost, protože nedochází k diskretizaci prostoru do jednotlivých buněk, jako v případě mřížek obsazenosti. Další výhodou je existence mnoha již hotových geometrických map pro řadu budov ve formě technické dokumentace. Tato dokumentace pak může být snadno upravena a poskytnuta robotu jako apriorní informace o daném prostředí.

2.2.3. Příznakové mapy

Pro určitá specifická prostředí s lokálně rozeznatelnými *příznaky* (tzv. *landmarky*), lze použít *příznakové mapy* [39]. Jako příznak by měl být použitý nějaký významný bod či objekt v prohledávané prostředí. Měl by jít snadno získat, být statický, ale zároveň i dostatečně specifický v daném prostředí [33]. Typickým příkladem mohou být například hrany (rohy) zdí, osamocené překážky, stromy ap.

Příznakové mapy uchovávají informace o pozici všech příznaků, včetně jejich nejistot. Pro každý příznak je tak poloha modelována jako rozdělení pravděpodobnosti. Kromě poloh příznaků je uchovávána také matice jejich vzájemných kovariancí, která představuje vzájemné relace mezi příznaky.

Podobně jako je mřížka obsazenosti limitována svojí celkovou velikostí, příznaková mapa je použitelná jen pro určité množství příznaků. S rostoucím počtem příznaků roste také kovarianční matice, která je při postupném budování mapy v každém kroku přepočítávána.

Příznaková mapa bývá vyžadována některými SLAM metodami, především metodami používajícími Kalmanův filter (viz část 4.1.1).

2.2.4. Další mapy

Mezi nejjednodušší senzorické mapy patří *bodová mapa*. Jedná se o pouhou množinu bodů, kde každý bod je reprezentován přesnými souřadnicemi. Body představují diskretizovaný povrch okolních překážek. Bodová mapa nijak nepostihuje nejistoty senzorických měření.

Topologické mapy poskytují naopak vysokou míru abstrakce. Narozdíl od map s nízkou mírou abstrakce není kladen důraz na přesné zaznamenání polohy jednotlivých překážek, ale spíše na vzájemné relace mezi nimi. To přirozeně vede na reprezentaci prostředí ve formě grafu, kde jsou významná místa v prostředí modelována jako uzly grafu a vzájemné hrany mezi uzly reprezentují vztahy mezi významnými místy (např. existence volné cesty mezi dvěmi volnými místy, viditelnost překážky ap.) [31].

Topologické mapy se hodí pro přibližnou lokalizaci polohy robotu ve velmi rozsáhlném prostředí a pro plánování tras robotu.

Symbolické mapy jsou modely prostředí s vysokou mírou abstrakce. Představují určité rozšíření topologických a geometrických map. Jednotlivým překážkám z geometrické mapy jsou přiřazena jména, která korespondují s jejich významem. Dále jsou definovány relace, které určují vlastnosti objektů. Symbolická jména jsou zadávána člověkem. Symbolické mapy nalézají využití např. při interakci člověka s robotem.

2.3. SLAM

Metody označované souhrně jako „*Simultání lokalizace a mapování*[3]“, zkráceně SLAM, řeší problém, který vyvstane, pokud budeme chtít autonomním mobilním robotem zmapovat neznámé prostředí:

- Aby mohl robot vytvářet inkrementálně mapu prozkoumávaného prostředí, je nezbytně nutné, aby měl k dispozici nějakou referenční informaci o svojí poloze.
- Robot naopak pro svoji lokalizaci potřebuje disponovat mapou neznámého prostředí.

Tento problém lze sice částečně obejít použitím nějakého externího lokalizačního systému (např. GPS, viz kapitola 3), nicméně tyto externí lokalizační systémy nemíváme zpravidla ve vnitřním prostředí k dispozici.

Simultální lokalizace a mapování tedy označuje proces, během kterého robot vytváří mapu neznámého prostředí a zároveň se ve stejné mapě lokalizuje, bez přímé podpory externích lokalizačních systémů.

2.3.1. Formální definice problému

Formálně můžeme SLAM problém definovat následujícím způsobem [37].

Uvažujeme lokalizaci robotu v rovině. Nechť t označuje *aktuální čas* a T čas konce činnosti robotu. Nechť x_t je *skutečná poloha robotu* v daném čase t. Vzhledem k předpokladu rovinného pohybu je dáno x_t jako vektor obsahující zpravidla dvě kartézské souřadnice a úhel natočení

[3] Z anglického: „Simultaneous Localization And Mapping“.

(orientaci) robotu. Jako *skutečnou trajektorii robotu* X_T pak budeme uvažovat posloupnost po sobě jdoucích poloh robotu v čase:

$$X_T = \{x_0, x_1, ...x_T\} \tag{2.3.1}$$

Dále je k dispozici nějakou informaci o relativním pohybu robotu, zpravidla odometrii (viz část 3.7.1). Nechť tedy u_t reprezentuje *odometrickou informaci* v čase $\langle t-1, t\rangle$. Jako *relativní pohyb robotu* U_T označíme posloupnost odometrických informací, která je dána jako:

$$U_T = \{u_1, u_2, ...u_T\} \tag{2.3.2}$$

Pokud bychom zde dosáhli absolutní přesnosti odometrického měření, postačovalo by to již plně k nalezení přesné koncové polohy robotu x_t. Bohužel, odometrické měření v praxi diverguje od správné polohy a je tedy nutné provádět korekci údajů o poloze [31].

Robot během svého pohybu provádí také měření dalšími senzorickými systémy. Nechť z_t je senzorické měření v čase t. Pak můžeme definovat posloupnost všech *provedených měření* jako:

$$Z_T = \{z_1, z_2, ...z_T\} \tag{2.3.3}$$

Nakonec definujeme model prostředí jako m. Počáteční poloha robotu x_0 je známá, ostatní polohy x_i jsou zatím neznámé. Pak lze definovat SLAM problém na základě teorie pravděpodobnosti vztahem:

$$P\left(X_T, m \mid Z_T, U_T\right) \tag{2.3.4}$$

Jinými slovy, hledáme nejpravděpodobnější skutečnou trajektorii robotu a model prostředí při znalosti senzorických pozorování a odometrických údajů. Takto formálně zavedený SLAM se také v literatuře často rozděluje do dvou skupin [37].

Tzv. *full SLAM* je definovaný přímo vztahem 2.3.4. Zde požaduje jako výstup kromě modelu prostředí co nejpravděpodobnější odhad celé skutečné trajektorie robotu. Tento SLAM může být teoreticky řešen také plně „offline“. Robot tedy může např. díky vzdálené teleoperaci pouze zaznamenat všechna senzorická měření, která jsou poté vyhodnocena některou ze SLAM metod.

Druhou variantou je tzv. *online SLAM,* kde požadujeme v každém čase t kromě modelu prostředí pouze nejpravděpodobnější odhad současné pozice x_t. Online SLAM lze formálně definovat jako:

$$P\left(x_t, m, \mid Z_T, U_T\right) \tag{2.3.5}$$

Tato varianta je zpravidla použita pro implementaci na autonomní mobilní robot.

Kapitola 3.

3 Senzorické systémy

Stejně jako jsou pro člověka klíčové jeho smysly, pro autonomní mobilní robot jsou zcela zásadní vhodné senzorické systémy. Bez senzorických systémů nemá robot možnost získávat samostatně jakékoliv informace z okolního prostředí.

V této kapitole se budeme soustředit pouze na senzory, které nalézají přímo uplatnění při procesu mapování či lokalizaci a pokusíme se srozumitelně a stručně uvést základní funkční principy a výhody a nevýhody jednotlivých senzorických systémů. Je brán ohled také na plánované použití na UAV.

3.1. Inerciální senzory

Mezi tzv. *inerciální senzory* řadíme gyroskopy a akcelerometry.

3.1.1. Gyroskop

Gyroskop je senzor měřící úhlovou rychlost. Zatímco v současnosti se pojem „gyroskop“ běžně používá také pro senzor náklonu, původní význam odkazuje na mechanické rotační setrvačníky, u kterých se uplatňuje tzv. gyroskopický efekt (Volný rotující setrvačník se snaží zachovávat osu své rotace díky svému momentu setrvačnosti, díky čemuž lze určit úhel natočení [25]). Tyto klasické mechanické gyroskopy se však především kvůli své vysoké hmotnosti a velkým rozměrům v mobilní robotice nepoužívají.

Nejpoužívanější a nejdostupnější alternativou jsou gyroskopy realizované MEMS[1] technologií. *MEMS gyroskopy* využívají Coriolisovou sílu, což je tzv. fiktivní síla, která působí na tělesa v rotující neinerciální vztažné soustavě (dále jen jako „rotující soustava“). Coriolisova síla závisí na rychlosti, se kterou se těleso vzhledem k rotující soustavě pohybuje a neuplatní se, pokud je

[1] Micro-Electro-Mechanical System označuje pokročilé technologie, které umožňují realizovat i složité mechanické zařízení na miniaturní úrovni a implementovat jej např. spolu s další pomocnou elektronikou na jeden čip [30].

těleso vůdči rotující soustavě v klidu, nebo se pohybuje rovnoběžně s osou rotace soustavy[7]. Coriolisova síla $\mathbf{F}_c$ je dána vztahem:

$$\mathbf{F}_c = m\mathbf{a_c} = 2m\omega \times \mathbf{v} \tag{3.1.1}$$

kde m je hmotnost tělesa $[kg]$, ω je úhlová rychlost rotující neinerciální vztažné soustavy $[rad \cdot s^{-1}]$, $\mathbf{v}$ je rychlost tělesa vzhledem k neinerciální vztažné soustavě $[m \cdot s^{-1}]$, $\mathbf{a}_c$ je tzv. Coriolisovo zrychlení $[m \cdot s^{-1}]$.

Dvě hmotná tělíska, většinou realizována jako vidličkový element, jsou rozkmitávána elektrostaticky na rychlost $\mathbf{v}$. Pokud se tato soustava zároveň otáčí úhlovou rychlostí ω začne působit na obě tělíska Coriolisova síla $\mathbf{F}_c$. To způsobí měřitelné výchylky obou tělísek ve směru kolmém na rovinu s buzenými kmity. Z velikosti výchylky je možné dopočítat hledanou úhlovou rychlost [30].

3.1.2. Akcelerometr

Akcelerometr je senzor měřící zrychlení, tedy obecněji síly působící na senzor. Akcelerometr je schopen měřit nejen zrychlení, ale také v důsledku např. vibrace či náklony.

Mechanický akcelerometr používá pro měření zrychlení malého hmotného tělíska - tzv. seismické hmotnosti. Seismická hmotnost je volně zavěšena v prostoru, tak, že umožněn její pohyb ve směru, ve kterém probíhá měření. Pokud začne na senzor působit nějaké zrychlení, je seismická hmotnost odpovídajícím způsobem vychylována. Za předpokladu ideální pružiny a viskózního tlumení[2] její polohu popisuje rovnice 3.1.2.

$$ma = m\ddot{x} + b\dot{x} + kx \tag{3.1.2}$$

kde x je okamžitá výchylka hmotného tělíska $[m]$, m je seismická hmotnost $[kg]$, a je zrychlení $[m \cdot s^{-1}]$, b je koeficient tlumení $[N \cdot s \cdot m^{-1}]$ a k je koeficient tuhosti pružiny $[N \cdot m^{-1}]$.

Ze známých parametrů senzoru a měřené výchylky x může být určena velikost působící síly. Nevýhodou mechanických akcelerometrů je velká náchylnost k rušení (vibracím) a potřeba rychlého vyčítání aktuální hodnoty výchylky. Podobně jako gyroskopy bývají v současnosti také akcelerometry vyráběny MEMS technologií.

Druhou skupinu tvoří tzv. *piezoelektrické akcelerometry*, které se využívají pro svoji činnost piezoelektrických vlastností některých materiálů. Seismická hmotnost zde stlačuje piezokrystal, na kterém vzniká elektrické napětí, jehož velikost je měřena a lze z ní určit velikost působící síly. Velkou výhodou těchto akcelerometrů je menší náchylnost na rušení vlivem vibrací. Na druhou stranu, vlastnosti piezoelektrických materiálů jsou teplotně závislé [30]. Existují i další

[2] Velikost viskózního tlumení je přímo úměrná rychlosti pohybu seismické hmotnosti[30].

řešení akcelerometrů, která různým způsobem kombinují či vylepšují výše uvedené základní principy [35].

Gyroskopy spolu s akcelerometry bývají vyráběny kromě samostatných senzorů také jako tzv. *inerciální měřecí jednotky*, zkráceně označované *IMU*. Tyto jednotky obsahují sdruženě trojici vzájemně kolmých gyroskopů a trojici akcelerometrů. Díky tomu lze určit plnou polohu měřeného tělesa v prostoru a což je výhodné pro potřeby mobilní robotiky. Pro robusnější odhady náklonů se provádí fúze dat z gyroskopů a akcelerometrů.

Jak však zmiňuje autor ve své práci [31], při určování polohy založené pouze na inerciálních senzorech diverguje velmi rychle poloha od správného řešení. Pro určování polohy robotu se tedy příliš nehodí a je nutné je používat ve spolupráci s dalšími senzorickými systémy. Na UAV se setkáváme především s využitím IMU jako senzoru pro odhadování náklonů celé platformy.

3.2. Sonar

Sonar je senzor pro měření vzdálenosti využívající odrazů vysílaných zvukových vln od okolních překážek. Na základě znalosti rychlosti zvuku je možné určit vzdálenost k nejbližší překážce. Jejich velkou výhodou je relativní jednoduchost a nižší cena. Na druhou stranu, jak uvádí autor [31], použití zvukových vln s sebou nese také řadu nevýhod.

Rychlost šíření zvukových vln je závislá na teplotě a tlaku prostředí. Pokud má být měření přesné v různých podmínkách, je nezbytné dalšími senzory provádět kompenzaci změn teploty a tlaku. Problematická je také různá odrazivost rozdílných materiálů.

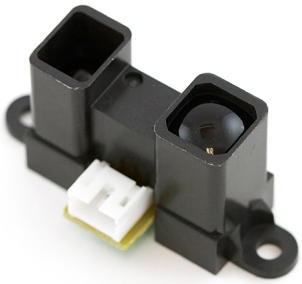
V oblasti robotiky tedy nalézají uplatnění především v systémech pro předcházení kolizím. Velké uplatnění nachází sonar také v prostředích, kde je použití optických senzorů vzdálenosti problematické (např. pod vodou, za sníženého osvětlení ap.).

Zajímavý sonar, speciálně vyvinutý pro použití na UAV, vyrábí firma MaxBotix (viz obrázek 3.1a). Tento senzor pracuje s frekvencí zvuku 42kHz, disponuje autokalibrací podle okolních podmínek a je dle manuálu velmi odolný k šumu a je schopen detekovat překážky až do vzdálenosti zhruba 7.5m. Měřená data jsou posílána po sběrnici I2C.

3.3. Proximitní senzory

Proximitní senzory jsou velmi jednoduché senzory pro detekci překážek v bezprostřední blízkosti robotu. Buď poskytují pouze informaci o existenci překážky, nabo v omezené míře (řádově desítkách centimerů) mohou také poskytovat informaci o vzdálenosti k překážce [23]. K tomu se používá jednoduchá triangulace.

(a) Sonar MaxBotix MB1240. Převzato z: `http://www.maxbotix.com`.



(b) IR proximitní senzor Sharp GP2Y0A02YK0F. Převzato z: `htt://www.sparkfun.com`

Obrázek 3.1.: Ukázky konkrétních senzorů

Patrně nejpoužívanější pro účely mobilní robotiky jsou senzory využívající infračerveného (dále jen IR) světla. Senzor obsahuje zpravidla dvojici IR vysílač - IR přijímač. IR vysílač přijímá odražené světlo a na základě přijaté intenzity světla je vyhodnocena existence překážky.

Ačkoliv je tento měřící princip vzdáleností relativně nepřesný, pro pouhou detekci existence překážky v bezprostřední blízkosti robotu a tedy pro nízkoúrovňové předcházení kolizím se hodí, ať už pro svoji celkovou jednoduchost, nízkou cenu, nebo nebo velmi malý proudový odběr.

Konkrétním příkladem lepších proximitních senzorů jsou senzory od firmy Sharp (viz obrázek 3.1b), které fungují na triangulačním principu. Senzory disponují analogovým napěťovým výstupem. Velikost výstupního napětí je úměrná měřené vzdálenosti. Firma vyrábí několik typů pro rozdílné vzdálenosti, až do vzdálenosti 150cm.

3.4. Laserový dálkoměr

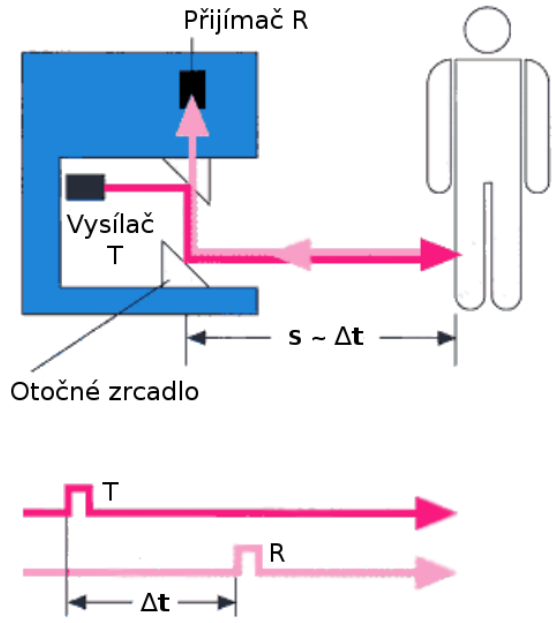
Laserové dálkoměry (laserové scannery, lidary) představují spolu s kamerami v současnosti dominantní senzory pro mapování vnitřního prostředí. Pro určení vzdálenosti od okolních překážek je používáno laserového paprsku, jehož odraz je snímán a vyhodnocován. Vzdálenost je určena na základě vyhodnocování fázového rozdílu, případně vyhodnocování doby letu.

Laserové dálkoměry se vyrábějí v různých provedeních, pro potřeby mobilní robotiky se však v součastnosti nejvíce hodí rotující laserové dálkoměry, které měří vzdálenosti k nejbližším překážkám, pravidla v jedné rovině. Jejich princip vysvětluje obrázek 3.2a.

Jeden tzv. *scan* je pak dán jako posloupnost dvojic „úhel natočení - vzdálenost k překážce“. Defacto jde tedy o polární souřadnice okrajů překážek vzhledem k poloze robotu.

Na trhu také existují 3D laserové scannery, které poskytují laserové scany z několika horizontálních rovin nad sebou. Ty se však pro aplikace na UAV příliš nehodí kvůli své velikosti, vysoké ceně i pomalejšímu procesu snímání. Novinkou v této oblasti je výrobek Puck LITE americké firmy Velodyne Lidar. Jedná se o laserový scanner, který snímá celé okolí (360° horizontálně, ±15° verikálně) s maximálním dosahem až 100m. Tento senzor je již přímo výrobcem určen i pro použití s UAV [1]. Vzhledem k vyšší váze (cca 590g) je však stále problematické použití tohoto laserového scanneru na menších kvadrokoptérách pro mapování vnitřních prostředí.

Velkou výhodou laserových scanerů obecně je dobrá přesnost. K často používaným laserovým dálkoměrům patří především výrobky od firem SICK a Hokuyo.



(a) Princip rotujícího laserového dálkoměru. Převzato z [31]

(b) Laserový dálkoměr Hokuyo URG-04LX, převzato z `http://dronesarefun.com`

Obrázek 3.2.: Laserové dálkoměry

3.5. Radar

Radar je senzor, který pracuje na podobném principu, jako laserový dálkoměr, místo světla - laseru, je však použito rádiových vln.

Radar, založený na původním principu (tzv. „*pulzní radar*“), který měří časový interval mezi vysláním a příjmem radiové vlny, se pro účely mobilní robotiky příliš nehodí. Jak uvádí ve své práci autor [10], se zmenšující se vzdáleností roste nutnost vyhodnocovat velmi krátké časové okamžiky a tedy i cena senzoru. Pro mobilní robotiku se více hodí tzv. *FMCW radar*[3], který

[3] Z anglického „Frequency Modulated Continuous Wave“

vysílá i přijímá nepřetržitě. V průběhu času je modulována frekvence signálu. Frekvence vysílaného signálu je porovnávána s frekvencí přijímaného signálu a vzdálenost od překážky je určena na základě znalosti použité modulace signálu. Radary tohoto typu jsou v současnosti již dostatečně kompaktní na to, aby bylo možné jejich nasazení v mobilní robotice. Ačkoliv bývá pro mapování vnitřního prostředí používáno především laserových dálkoměrů a kamer, využití radaru může být zajímavou alternativou či doplněním do určitých specifických prostředí, především pro svůj delší dosah.

Další výhodou může být použití rádiových vln - elekromagnetického vlnění s podstatně nižší frekvencí. Rádiové vlny se tak především lépe šíří překážkami. Získaná radarová data tak narozdíl od měření z laserového dálkoměru mohou poskytovat do jisté míry více informací o předkážkách, protože se netýkají pouze obrysů překážek. To samé ale může být zároveň nevýhodou. Kromě toho mají radary výrazně širší „vyzařovací lalok“ a menší úhlové rozlišení.

Radar tedy může být přínosným senzorickým systémem pro mapování velkých prostor, ale pro mapování běžného vnitřního prostředí se vzhledem ke svému úhlovému rozlišení a přesnosti příliš nehodí. Problematikou využití radaru pro lokalizaci a mapování v mobilní robotice se velmi komplexně zabývají autoři v publikaci [27].

3.6. Kamera

Použití kamer je v mobilní robotice v současnosti velmi rozšířené. Kromě pouhého záznamu či přenosu videa (např. za účelem teleoperace robotu) se kamery používají často také pro měření vzdálenosti. Tyto metody úzce souvisejí s počítačovým viděním.

Jednou z variant pro měření vzdáleností je použití tzv. *stereo kamer* [31]. Jde o dvojici (nebo i větší množství) kamer, která snímají takřka totožnou scénu každá z trochu jiné pozice. Tyto kamery jsou vzájemně pevně zafixovány a je známá jejich vzdálenost. Poté, co všechny kamery sejmou scénu, jsou v obrázcích z jednotlivých kamer nalezeny vzájemně si korespondující významné body. Následnou triangulací je odhadnuta vzdálenost jednotlivých bodů scény. Výsledkem může být sestavení tzv. hloubkové mapy. Tento způsob určování vzdálenosti je inspirovaný zrakem živočichů - existencí páru očí. Ačkoliv tento systém dává především na kratší vzdálenosti dobré výsledky, je celý proces relativně náročný na výpočetní výkon. Příkladem konkrétního produktu je stereokamera ZED od americké firmy Stereolabs Inc., která je velmi kompaktní, disponuje připojením přes USB 3.0 a poskytuje např. fullHD video se vzorkovací frekvencí až 30 snímků za sekundu. Dodávanými knihovnami je možné ze stereo snímků dopočítat hloubkové mapy.

Další variantou jsou kamery využívající tzv. *aktivní triangulaci*. Do scény, snímané kamerou, je promítán laserem světlý bod či proužek. Z polohy laserového paprsku ve snímaném obraze a známé vzdálenosti mezi laserem a kamerou je určena triangulací vzdálenost od překážky. Vzhledem k pomalejší rychlosti snímání nalézá tato technologie uplatnění především v průmyslu např.

vytváření při 3D modelů objektů a pro mobilní robotiku se však příliš nehodí [31].

Záležitostí posledních let jsou kamery pracující s odraženým světlem a dobou letu světla, někdy také označované jako *RGB-D kamery* [18]. Jde vlastně o kombinaci principů použitých u laserových dálkoměrů a kamery. Celá scéna je mžikově osvícena buď laserem, nebo IR zářením. Pro každý pixel je kromě klasických obrazových dat měřena doba letu (resp. fázový posun) světla. Podle toho lze sestavit hloubkovou mapu snímané scény. Do této kategorie spadá např. zařízení Kinect od firmy Microsoft, které disponuje rozlišením 640×480 pixelů a pro osvětlování scény používá modulované IR záření. Výhodou těchto systémů je velká rychlost snímání, a také to, že nevyžadují výpočetní výkon pro určení hloukové mapy scény podobně jako stereo kamery, nevýhodou pak nízké rozlišení ve srovnání s klasickými či stereo kamerami.

Jiný způsob využití kamery publikovali autoři v práci [19], ve které navrhli hardwarový modul Px4flow pro mobilní roboty, především pak pro UAV, který je schopen odhadovat pohyb robotu na základě vyhodnocování optického toku z kamery. Modul dále obsahuje také sonar, který poskytuje informaci o výšce UAV nad zemí. Celý modul byl navrhován speciálně pro vnitřní prostředí s horším osvětlením a může tedy posloužit jako videoodemetrie - náhrada klasické odometrie (viz část 3.7.1) i pro UAV ve vnitřním prostředí. Modul disponuje bohatou konektivitou. Lze ho připojit přes USB, nebo na sběrnice UART či I2C. Pro komunikace po USB je používám Mavlink protokol [3]. Modul je schopen poskytovat informace o rychlostech ve směru dvou horizontálních os a výšce s frekvencí až 250Hz [19].

(a) Modul PX4FLOW, převzato z http://pixhawk.org. (b) Stereokamera ZED, převzato z www.stereolabs.com

Obrázek 3.3.: Příklady použití kamer

3.7. Další senzorické systémy

3.7.1. Odometrie

Mezi základní způsoby odhadu pozice mobilních pozemních robotů patří tzv. *odometrie.* Odometrií je zde myšleno odhadování polohy robotu na základě znalosti měření dat z aktuátorů a modelu robotu. Typicky jsou měřeny otáčky pohonů robotu inkrementálními senzory. Na základě kinematického modelu robotu lze pak určit rychlost robotu a integrací získat jeho polohu.

Odometrie u kolových robotů představuje velmi jednoduchou a běžně používanou metodu, jak získat odhad pozice robotu v reálném čase. Na druhou stranu, s uraženou vzdáleností roste chyba odhadu pozice. Přesnost odometrie je sice úměrná rozlišení inkrementálních senzorů, ale i při použití přesných senzorů hrají roli také další vlivy (např. robotu může podklouznout kolo ap.). Obecně se dá říci, že pro přesné určení pozice robotu se v běžných podmínkách nelze spoléhat pouze na odometrii [31].

Bohužel, v případě použití UAV nemáme klasickou odometrii k dispozici. Tento nedostatek lze částečně kompenzovat použitím videoodemetrie, zmíněné v pokapitole 3.6.

3.7.2. Globální družicový polohový systém

Globální družicový polohový systém je služba, která umožňuje určit polohu na zemském povrchu či ve vzduchu díky systému družic, které obíhají zemi po známých drahách. Tyto družice obsahují velmi přesné atomové hodiny a neustále vysílají informace o svojí poloze a přesný čas. Pokud pasivní přijímač přijímá signál alespoň od 3 družic, je možné určit jeho polohu na zemském povrchu. Pokud je viditelných družic více, lze navíc určit i nadmořskou výšku a celkově určenou polohu dále zpřesnit. Do této oblasti spadají družicové systémy jako jsou americký Navstar GPS a ruský GLONASS, případně vyvíjené systémy jako evropský Galileo nebo čínský Baidu.

Bohužel, ačkoliv jsou tyto systémy výborné pro lokalizaci robotů ve venkovním prostředí (jejich kombinací lze docílit na volném prostranství přesnosti řádově metrů), ve vnitřním prostředí není signál k dispozici vůbec, nebo je přesnost určené polohy nedostatečná.

Na podobném principu může být založen i systém pro lokální lokalizaci robotu ve vnitřním prostředí (místo družic jde o pevně umístěné vysílače), to ale opět nepřipadá v úvahu při mapování neznámého vnitřního prostředí.

Kapitola 4.

4 SLAM metody

Jak již bylo předesláno v podkapitole 2.3, vyřešení SLAM problému je klíčové i pro realizaci mapování vnitřního prostředí UAV. V této kapitole nejprve zběžně představíme nejpoužívanější obecné přístupy k vyřešení SLAM problému. Na závěr této kapitoly uvedeme stav řešení mapování vnitřního prostředí UAV ve světě 4.2.

Podrobný rozbor různých přístupů k řešení SLAM problému není předmětem této práce, ale lze dohledat například v literatuře [12, 36, 37].

4.1. Obecný přehled metod

4.1.1. Metody s Kalmanovým filtrem

Metody s Kalmanovým filtrem (označované jako EKF-SLAM) patří mezi tradiční SLAM metody. Jsou založené na získávání příznaků z měřených dat (viz část 2.2.3).

Kalmanův filtr je algoritmus, který umožňuje predikovat neznámé hodnoty stavů diskrétního systému na základě znalosti modelu systému a výstupů systému, které jsou ovlivněné bílým šumem[1] [13]. Rozšířená verze tohoto filtru pracuje stejně, ale i pro nelineární systémy. Toho se využívá pro odhadování pozice při řešení SLAM problému. Poloha robotu je modelována normálním rozdělením. Úloha nalezení nejpravděpodobnější polohy robotu tedy odpovídá nalezení střední hodnoty daného normálního rozdělění a s příslušnou nejistotou - rozptylem. Díky pravděpodobnostnímu přístupu k poloze je možné postihnout neurčitosti senzorů i vytářeného modelu prostředí. Algoritmus pracuje velmi zjednodušeně ve dvou krocích: predikci a aktualizaci [31].

- Nejprve je odhadnuta na základě odometrie a známých nejistot předpokládaná pozice robotu (a její nejistota).

[1] Bílý šum je šum (rušení signálu) s normálním rozdělením. Jinými slovy, všechny frekvence signálu, ke kterému je superponován šum, jsou rušeny stejně.

- V následujícím kroku je odhad pozice aktualizován na základě senzorického měření. Přitom se pracuje s tzv. váhovacími koeficienty. Měření s menší nejistotou má pak větší váhu díky většímu váhovacímu koeficientu. Rozpoznané příznaky jsou ukládány do příznakové mapy.

Velkou výhodou EKF přístupů je skutečnost, že uchovávání rozložení pravděpodobností poloh samotných příznaků vyžaduje podstatně méně paměti, než uchování surových senzorických dat. Přesto se tento přístup nehodí do rozlehlejších prostředí. Mázl [31] ve své práci uvádí, že výpočetní složitost roste kvadraticky s počtem příznaků. Je to dáno tím, že Rozšířený Kalmanův filter potřebuje při každé iteraci algoritmu přepočítat matici vzájemných kavariancí (tzv. Gaussián) všech příznaků. V důsledku to dovoluje tuto metodu použít pouze pro relativně malá prostředí s několika stovkami příznaků. S použitím příznaků je také spojena další nepříjemná skutečnost, nutnost implementce nějakého algoritmu, který určuje korespondující dvojice příznaků z příznakové mapy a aktuálního senzorického měření. V případě nekorektního spárování příznaků hrozí divergence celé lokalizace od správné polohy. Navíc algoritmus umožňuje použít pouze normální rozložení pravděpodobnosti.

4.1.2. Metody s Informačním filtrem

Další alternativou jsou přístupy vycházející z tzv. *Monte-Carlo lokalizace*, tento přístup je někdy označován také jako tzv. *Informační filter*[2]. Základní myšlenkou této globální lokalizační metody je modelovat polohu robotu tzv. *particly*[3]. Particly jsou vzorky reprezentující diskretizovanou distribuci pravděpodobnosti polohy robotu. Algoritmus Monte-Carlo lokalizace se pokouší manipulací s particly dosáhnout toho, aby vzniklá diskrétní distribuce tvořená particly odpovídala skutečné spojité distribuci pravděpodobnosti. Toho se dosahuje tzv. *importance faktory*, což jsou koeficienty uvádající váhu jednotlivým vzorkům. Jeden particle tedy odpovídá určité jedinečné poloze a jeho importance faktor vyjadřuje pravděpodobnost, se kterou se na tomto místě robot nachází. Princip Monte-Carlo lokalizace je následující [31]:

Na začátku algoritmu se vygeneruje nějaký konkrétní počet particlů. Se stejnou pravděpodobností se může robot nacházet na kterémkoliv z nich. Jakmile se robot začne pohybovat, začně také získávat odometrická a senzorická měření. Pak se rekurzivně odehrávají následující dva kroky algoritmu:

- Při tzv. *predikci* dojde k posunu všech particlů a jejich importance faktorů podle odometrické informace. Podle přijatého měření se nastaví importance faktory. Pravděpodobnějším patriclům jsou zvýšeny a naopak.
- Při tzv. *korekci* se upravuje celá množina particlů i jejich importance faktorů. Dojde k *převzorkování* podle importance faktorů na základě shody senzorického měření s predikcí.

[2] Z anglického „Particle filter“.
[3] Z anglického „particles“.

Nejméně pravděpodobné particly jsou vypuštěny a nahrazeny novými particly, které jsou vygenerovány v oblastech s vysokou pravděpodobností výskytu robotu.

Na základě této lokalizační metody a principu klasického EKF-SLAM algoritmu navrhli autoři [29] tzv. FastSLAM. Klíčovou myšlenkou FastSLAM algoritmu je dekompozice klasického SLAM problému na odhad posteriorních pravděpodobností trajektorií robotu $x_{1:t}$ a K odhadů pozic příznaků, které jsou podmíněny odhady trajektorií robotu. To je možné, protože platí faktorizace 4.1.1 [31].

$$p(x_{1:t}, \theta \mid z_{1:t}, u_{1:t}, n_{1:t}) = p(x_{1:t} \mid z_{1:t}, u_{1:t}, n_{1:t}) \prod_k p(\theta_k \mid x_{1:t}, z_{1:t}, u_{1:t}, n_{1:t}) \tag{4.1.1}$$

kde $x_{1:t}$ představuje pozice robotu, $z_{1:t}$ jsou senzorická měření, $u_{1:t}$ je odometrie robotu, θ jsou pozice příznaků a $n_{1:t}$ značí známé korespondence příznaků, K je počet příznaků. Vše v časovém intervalu $\langle 0; t \rangle$.

Odhad posteriorních distribucí $p(x_{1:t} \mid z_{1:t}, u_{1:t}, n_{1:t})$ je realizován upraveným Informačním filtrem. Ke každému particlu reprezentujícímu možnou polohu robotu v čase t jsou aktualizovány pozice příznaků Rozšířeným Kalmanovým filtrem.

Hlavní předností FastSLAM algoritmu je skutečnost, že při stejném počtu příznaků je výrazně méně výpočetně náročný (místo jednoho obrovského Gaussiánu je při každém kroku aktualizováno více rozměrově malých Gaussiánů). Algoritmus FastSLAM byl autory úspěšně otestován i ve velmi rozlehlém prostředí se zhruba 50,000 příznaky [37].

4.1.3. Scan-matching metody

Scan-matching metody přistupují k SLAM problému bez použití příznaků[4]. Základní myšlenkou těchto metod je nalezení optimální transformace mezi dvěmi (či několika) následujícími senzorickými měřeními, případně je hledána transformace např. mezi mřížkou obsazenosti a novým senzorickým měřením. Optimální transformace je pak hledána nějakou optimalizační metodou, většinou optimalizačním algoritmem nejmenších čtverců [31].

Výhoda těchto metod je zřejmá. Vzhledem k faktické absenci příznaků je použití těchto metod jednodušší v prostředích, kde je extrahování a hledání vzájemných asociací příznaků komplikované. Také je relativně jednoduchá tvorba primitivní bodové mapy. Na druhou stranu jsou tyto metody citlivé na počáteční podmínky. V případě, že dvě sousední senzorická měření (popř. více měření) jsou od sebe příliš vzdálená a nepodaří se získat kvalitní transformaci může dojít k divergenci od správného řešení.

[4] Z jiného úhlu pohledu lze považovat každý bod senzorického měření za samostaný příznak a pak se tento rozdíl vytrácí.

4.1.4. Graf-SLAM

Další skupina metod vychází z myšlenky, že řešení SLAM problému lze pojmout jako globální optimalizační problém.

Pohyb robotu je modelován jako graf, jehož vrcholy reprezentují buď polohy robotu v prostředí v čase t nebo pozorované příznaky. Hrany mezi jednotlivými vrcholy představují omezující podmínky[5]. Omezující podmínky jsou tvořeny buď odometrickou informací (odhad vzdálenosti mezi dvěmi sousedními vrcholy s polohou robotu), nebo informací o vzdálenostech k příznakům (mezi dvěmi vrcholy: poloha robotu - příznak) [37].

Dále se vychází z myšlenky, publikované autory [28], že graf vytvořený podle těchto pravidel lze považovat z jistého úhlu pohledu také za model mechanického systému pružin. Výpočet stavu tohoto systému s minimální energií pak dle této analogie odpovídá vyřešení SLAM problému. Dále lze odvodit, že vztah formulující fullSLAM problém 2.3.4 lze upravit do takového tvaru, že lze na problém aplikovat běžně používané optimalizační metody. Díky tomu lze SLAM problém řešit jako globální optimalizační problém nad vytvořeným grafem [37].

Vzhledem ke globální optimalizaci se jedná zpravidla o „offline“ SLAM, který vyžaduje mít k dispozici záznam veškerých dat. Zde však tato metoda poskytuje velmi dobré výsledky a zvládá i tvorbu map pro velmi rozlehlá prostředí.

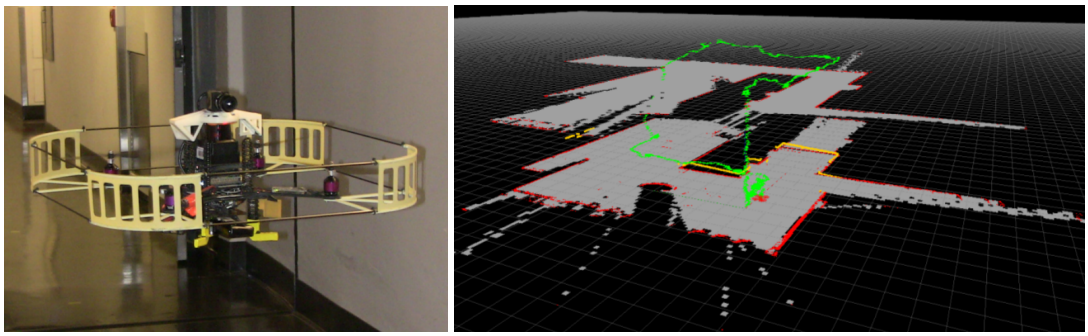
4.2. Stav problematiky mapování vnitřního prostředí UAV ve světě

Zatímco mapování vnitřního prostředí plně autonomními pozemními roboty má své počátky už v 80. letech minulého století [36], v civilním sektoru bylo realizováno autonomní mapování neznámého prostředí UAV až v 90. letech [32]. Autoři zde předpokládali venkovní prostředí a pro lokalizaci využívali data z několika nezávislých kamer a IMU. Celková miniaturizace UAV začala postupně umožňovat také aplikace ve vnitřním prostředí. Jedny z prvních publikovaných prací zabývajících se autonomním mapováním vnitřního prostředí byly [21, 26]. V této podkapitole rozebereme několik různých přístupů k této problematice publikovaných během posledních deseti let.

Autoři [14] realizovali primitivní autonomní 2D lokalizaci a mapování vnitřního prostředí pouze za použití levných a běžně dostupných senzorů (čtveřice sonarů, dvojice proximitních senzorů, IMU). Kromě samotného SLAM algoritmu řeší autoři také návrh regulátoru pro řízení helikoptéry. Dále autoři přijali zjednodušující předpoklad, že mapované prostředí je složené pouze z prostor ve tvaru obdélníků. Díky tomu je možné ukládat model světa pouze ve zjednodušené

[5] Z anglického „constraint“.

parametrické formě a veškeré výpočty zvládá zabudovaný mikrokontrolér. Autonomní helikoptéra má definvanou sadu pravidel, podle kterých se pohybuje, hledá volná místa a překážky a vytváří inkrementálně primitivní mapu prostředí.



(a) Specializovaná kvadrokoptéra od firmy Ascending Technologies, GmbH.

(b) Mapa dvoupatrového domu se zaznamenanou trasou kvadrokoptéry (zeleně).

Obrázek 4.1.: Mapování vnitřního prostředí MAV, převzato z [24].

Sofistikovanější přístup je uveden v práci [24]. Zde kolektiv autorů publikoval autonomní způsob prozkoumávání neznámého obecného 3D prostředí s požadavkem co nejlepší kvality lokalizace a mapování, ale zároveň také realizovatelnosti všech výpočtů na HW vybavení přímo na kvadrokoptéře (viz obrázek 4.1a). Základní zdroj senzorických dat tvoří 2D laserový dálkoměr, doplněný kamerou, IMU (s magnetometrem a barometrem) a relativně výkonným počítačem. Inkrementální ICP SLAM algoritmus (viz část 5.3.1) autoři vylepšují „pasováním“ nových laserových scanů na existující mřížku obsazenosti. Pro lepší detekci návratu[6] MAV na již navštívené místo autoři navíc používají obrázků z kamery, ze kterých jsou extrahovány příznaky a tyto ukládány. Kromě toho je zároveň udržován také aktuální graf poloh (podobně jako je zmíněno v části 4.1.4). Díky průběžné globální optimalizaci grafu a hledání korespondujících příznaků z kamerových dat je možné získat kvalitnější a z globálního hlediska konzistentnější mapu prostředí, než by dovoloval samotný ICP SLAM algoritmus (viz obrázek 4.1b).

Také v práci [38] byl jako hlavní senzor použit laserový dálkoměr, ale místo klasické kamery pro snímání obrazu byla použita kamera směřující k podlaze pro výpočet optického toku a barometr pro určení přibližné výšky z atmosférického tlaku. Autoři implementovali upravený FastSlam algoritmus, popisovaný obecně v 4.1.2. Z optického toku z kamery je integrován odhad polohy kvadrokoptéry, který slouží jako náhrada klasické odometrie pro FastSLAM algoritmus. Autoři používají příznakovou mapu a předpokládají pouze 2D mapování (v jednom patře budovy).

Místo laserového dálkoměru lze použít jako hlavní zdroj senzorických dat také RGB-D kameru [34]. V této publikaci je využíván tzv. RGB-D SLAM, což je SLAM algoritmus rozšíře-

[6] V anglicky psané literatuře je tento jev označován jako „loop closure“.

jící myšlenky klasického 2D inkrementálního scan-matching algoritmu do 3D. Místo laserových scanů jsou hledány transformace mezi obecnějšími shluky bodů (hloubkovými mapami) ve 3D. Na hloubkové mapy mohou být poté namapovány textury, tedy klasická kamerová data. Dále může být také urdržován graf s pozicemi robotu a optimalizačními technikami může být celá mapa zlepšena, podobně jak bylo popsáno v části 4.1.4.

Alternativu pro vytváření 3D modelu prostředí přinášejí autoři [20] v podobě HW modulu, který obsahuje kameru, IMU a laserový dálkoměr upevněný na otočném závěsu. Laserový dálkoměr poskytuje informaci o vzdálenostech v celé snímané scéně a v důsledku tedy hloubkovou mapu. Pro řešení SLAM problému byl využit EKF-SLAM algoritmus.

Jako další práce, které se zbývají problematiku implementace autonomního mapování vnitřního prostředí UAV lze uvést např. [16, 17].

Kapitola 5.

5 Implementované řešení mapování

V této kapitole se komplexně zabýváme návrhem a implementací mapování vnitřního prostředí UAV. V pokapitole 5.1 nejprve popíšeme použité hardwarové vybavení a definujeme vhodná omezení a předpoklady pro vyřešení úlohy (viz podkapitola 5.2). Dále v podkapitole 5.3 diskutujeme výběr metody mapování. Ve zbylých částech této kapitoly rozebíráme implemetované softwarové řešení. Na závěr uvádíme také nejdůležitější možnosti nastavení programu pro případné uživatele bez znalosti zdrojového kódu.

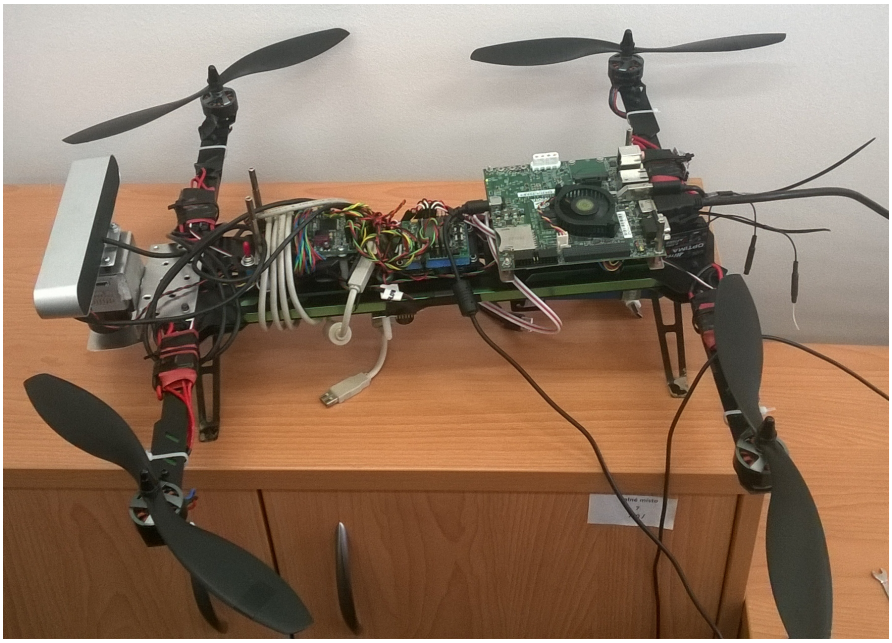
5.1. Hardwarové prostředky

Pro realizaci mapování byla sestavena kvadrokoptéra z běžně prodávých modelářských dílů, která byla doplněna o vhodné senzorické systémy a další komponenty s ohledem na běh programů pro mapování a případné budoucí plně autonomní chování (viz obrázek 5.1). Vzájemnou datovou komunikaci mezi jednotlivými subsystémy shrnuje obrázek 5.2, napájení jednotlivých subsystémů je patrné z obrázku 5.3.

5.1.1. Kvadrokoptéra

Mechanickou nosnou konstrukci tvoří montovaný rám z hliníkových profilů a sklolaminátu HobbyKing H4. Čtveřice BLDC motorů AX-2810Q-750KV (max. proud 30 A, maximální napětí 15 V) je osazena vrtulemi s rozpětím 304mm. Elektrickou komutaci zajišťují regulátory Turingy PLUSH-40A.

O řízení letu (přímé ovládání regulátorů motorů) a základní stabilizaci letu se stará RC řídicí jednotka HobbyKing KK 2.1.5. Jednotka obsahuje vlastní IMU a disponuje jednoduchým uživatelským rozhraním s LCD displayem, které umožňuje např. snadné nastavení konstant používaných PID regulátorů pro stabilizaci při letu. Řídicí signály z RC vysílačky (signály vzdálené teleoperace) jsou přijímány na RC přijímači HiTEC Optima 9, který pracuje v pásmu 2.4GHz.



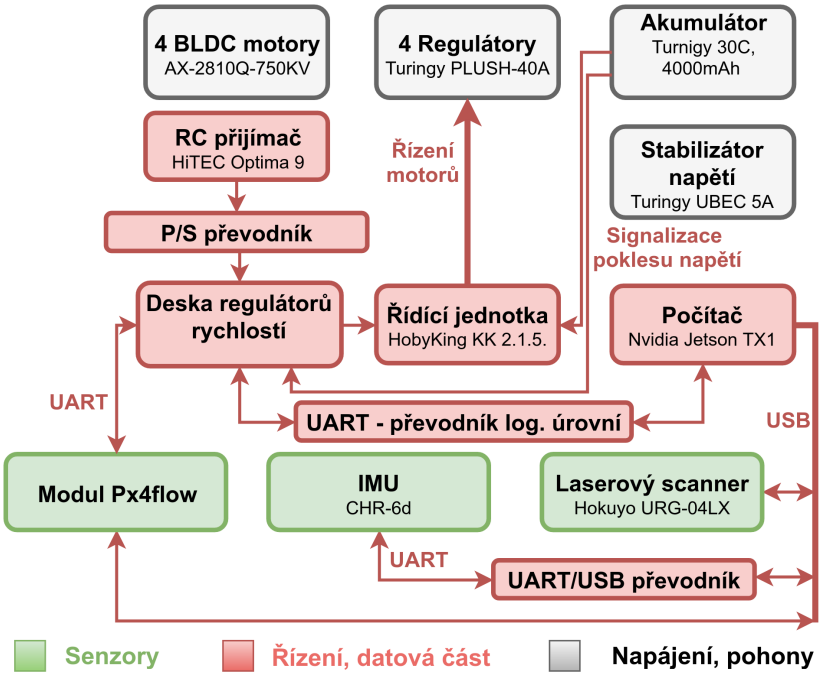
Obrázek 5.1.: Realizovaná kvadrokoptéra s instalovanými senzorickými systémy. Stereokamera ZED není v této práci použita.

Výstupem přijímače je 9 paralelních signálů, které mohou být využity kromě běžného řízení letu také pro další účely. Tyto signály jsou poté pomocným obvodem převedeny na sériový přenos (obvod je označovaný v obrázku 5.2 jako „P/S převodník“). Signál je modulovaný PPM[1] modulací, která je běžně používána v RC technice a takto modulovaný signál umí zpracovávat i řídicí jednotka KobbyKing KK 2.1.5 [11].

Dále je signál zpracováván „Deskou regulátorů rychlostí“, kterou navrhl ve své práci autor [11]. Tato deska představuje mezistupeň nutný pro budoucí autonomní let kvadrokoptéry. Díky ní je možné plně autonomní let či sledování definované trajektorie, deska však zároveň zachovává možnost vzdálené teleoperace. Jednotka implementuje stabilizaci kvadrokoptéry založenou především na datech z IMU a modulu Px4flow PID regulátorem a volitelně umožňuje také implementaci lepšího prediktivního regulátoru. Navíc deska disponuje bohatou konektivitou a je schopna komunikovat po sériových sběrnicích s dalšími senzorickými systémy (např. ZigBee modul pro telemetrii ap.).

Prostředí pro běh algoritmů mapování, lokalizace a případného plánování tras představuje kompaktní počítač Nvidia Jetson TK1. Tento počítač je navržen speciálně pro vývoj vestavěných aplikací v oblastech robotiky a počítačového vidění. Disponuje procesorem ARM Cortex™- A15, operační pamětí 2GB RAM, grafikou NVIDIA Kepler (192 CUDA jader), flash úložištěm 16GB a bohatou konektivitou. Jako operační systém je používána linuxová distribuce Ubuntu. Mož-

[1] Pulsně polohová modulace je způsob implusní modulace, kde je okamžitá hodnota modulačního signálu vyjádřena časovou polohou pulsu v daném rámci [40].



Obrázek 5.2.: Datová komunikace subsystémů kvadrokoptéry.

nost sériové komunikace počítače Jetson TK1 a Desky regulátorů rychlostí poskytuje výborný předpoklad pro budoucí plně autonomní činnost kvadrokoptéry.

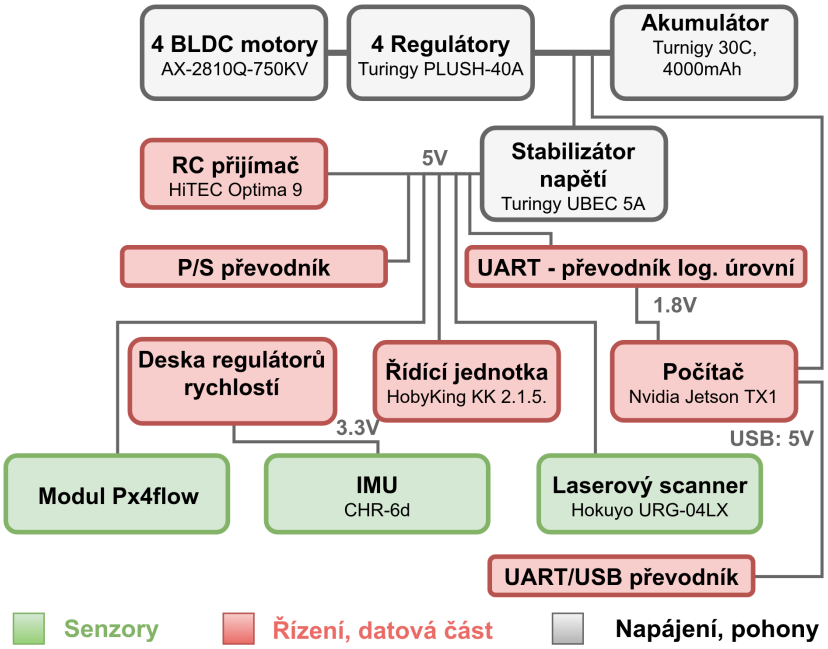
Pro vzájemnou komunikaci bylo využito rozhraní UART. Vzhledem k tomu, že počítač Jetson TK1 i Deska regulátorů obsahují rozhraní UART s rozdílnými napěťovými úrovněmi, vytvořili jsme pro tento účel převodník logických úrovní, jehož schema zapojení je na obrázku 5.4.

Zdroj energie pro veškeré subsystémy tvoří 3-článkový Li-Pol akumulátor Turingy 30c s kapacitou 4000mAh. Kromě počítače a regulátorů s motory je veškerá elekronika napájena ze stabilizátoru napětí Turingy UBEC 5A., který poskytuje stabilní stejnosměrné napětí 5 V, s maximálním proudovým odběrem 3 A. Z Desky regulátorů rychlostí je také vyvedeno napětí 3.3 V pro napájení IMU. Akumulátor disponuje také signalizačním výstupem, který umožňuje sledovat stav napětí na akumulátoru. Tento výstup je připojen přímo na jednoúčelový RC alarm, který akusticky i vizuálně signalizuje uživateli blížící se vybití baterie (tento alarm není zahrnut ve schematech 5.3 a 5.2). Kromě toho je hodnota výstupního napětí akumulátoru předávána také do Řídicí jednotky a Pomocné řídicí desky. Pro pohodlnou obsluhu byl instalován také přepínač, který umožňuje odpojení napájení řídicí elektroniky.

5.1.2. Senzorické systémy

Ze všech dostupných senzorů pro tento experiment byly vybrány následující senzory.

2D laserový dálkoměr Hokuyo URG-04LX představuje hlavní zdroj senzorických dat pro ma-



Obrázek 5.3.: Napájení subsystémů kvadrokoptéry.

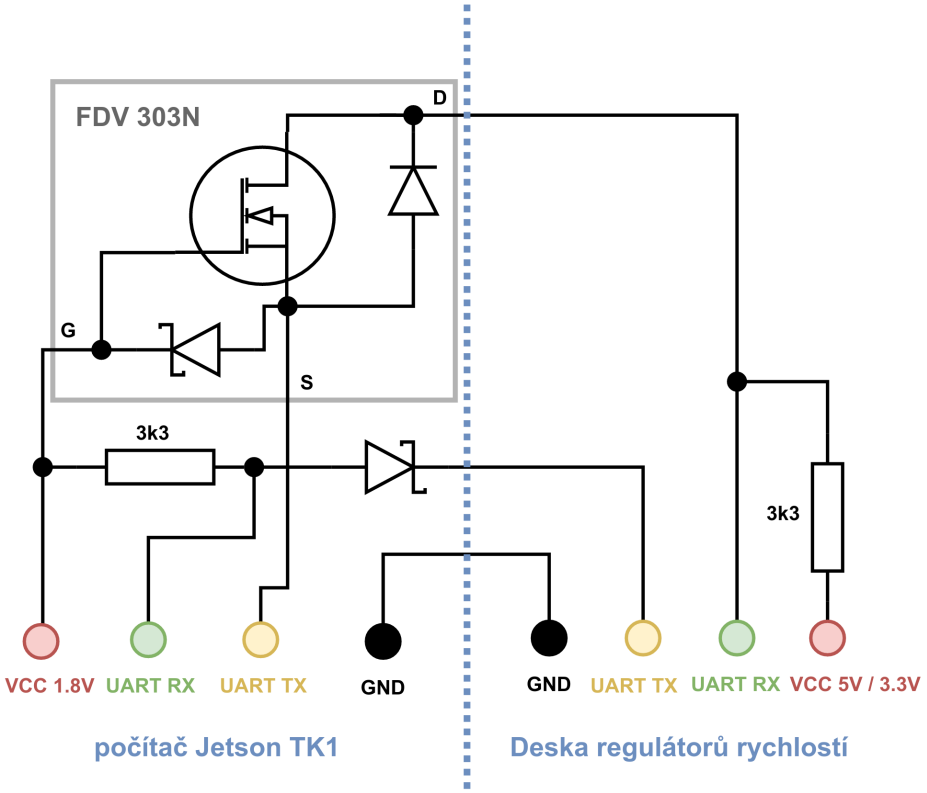
pování. Tento laserový dálkoměr je přímo navržen pro robotické aplikace ve vnitřním prostředí. Výhodou je dostupnost svobodných ovladačů pro linuxové operační systémy a nižší cena. Naopak hlavním nedostatkem oproti dražším modelům je výrazně nižší dosah i nižší snímací frekvence. Nejdůležitější parametry senzoru shrnuje tabulka 5.1.1.

Protože prvotní experimenty ukázaly, že vybraný algoritmus mapování za použití samotného laserového dálkoměru může selhat v určitých typech prostředí (viz část 6.1), byly zvoleny ještě další podpůrné senzory.

Pro robusní odhadování orientace kvadrokopréry v prostoru byla použita **inerciální měřící jednotka CHR-6d** od firmy CH Robotics. Kromě 3-osého akcelerometru a 3-osého gyroskopu obsahuje jednotka také mikrokontrolér, který implementuje zpracování dat ze senzorů, sériové rozhraní UART a Kalmanův filtr, který provádí fúzi dat z gyroskopů a akcelerometrů a robusnějším způsobem odhaduje přímo úhly náklonu. IMU komunikuje přes převodník UART/USB s počítačem.

Jako náhrada klasického odometrického měření byl použit **modul Px4flow**, podrobněji rozebíraný již v části 3.6. Modul je připojený přes USB do počítače, ale zároveň také přes rozhraní UART do Desky regulátorů rychlostí.

IMU a modul px4flow byly připevněny na spodní část kvadrokoptéry, pro laserový dálkoměr byla vytvořena podpůrná konstrukce pro připevnění na čelní části kvarkokoptéry. Orientace vztažné souřadné soustavy je patrná ze obrázku 5.6. Ve smyslu této soustavy je určována pozice při mapování prostředí.

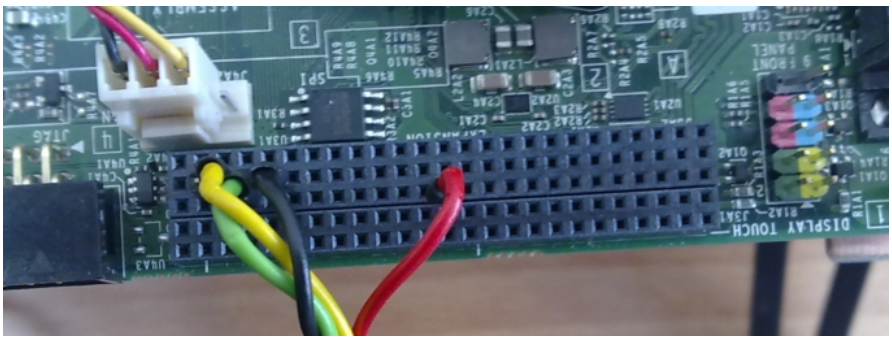


Obrázek 5.4.: Převodník logických úrovní pro sériovou komunikaci mezi počítačem Jetson TK1 a Deskou regulátorů rychlostí. Barvy ve schematu odpovídají barvám použitých reálných vodičů.

5.2. Omezující podmínky

Jak již bylo popsáno v kapitolách 2, 4, mapování neznámého prostředí je velmi komplexní problém. Při mapování vnitřního prostředí popsanou kvadrokoptérou proto přijmeme některé zjednodušující předpoklady, které redukují nároky na výpočetní výkon potřebný pro běh algoritmu a mohou být i nezbytné pro realizovatelnost s daným senzorickým vybavením.

- Předpokládáme pouze vnitřní prostředí budov. Stěny místností svírají pravý úhel s horizontální rovinou (podlahou).
- Kvadrokoptéra se pohybuje přibližně v rovině rovnoběžné s podlahou. Při mapování je vytvářena 2D mapa prostředí, která odpovídá půdorysu vnitřního prostředí.
- Celé mapované prostředí je statické. V průběhu mapování se tedy v dosahu senzorických systémů kvadrokoptéry nevyskytují žádné objekty, které by měnily svoji polohu (lidé, zvířata ap.).
- Materiál stěn a překážek v prostředí poskytuje dostatečně dobrý odraz pro laserový paprsek použitého dálkoměru a intenzita osvětlení prostředí je dostatečná pro použití modulu px4flow.



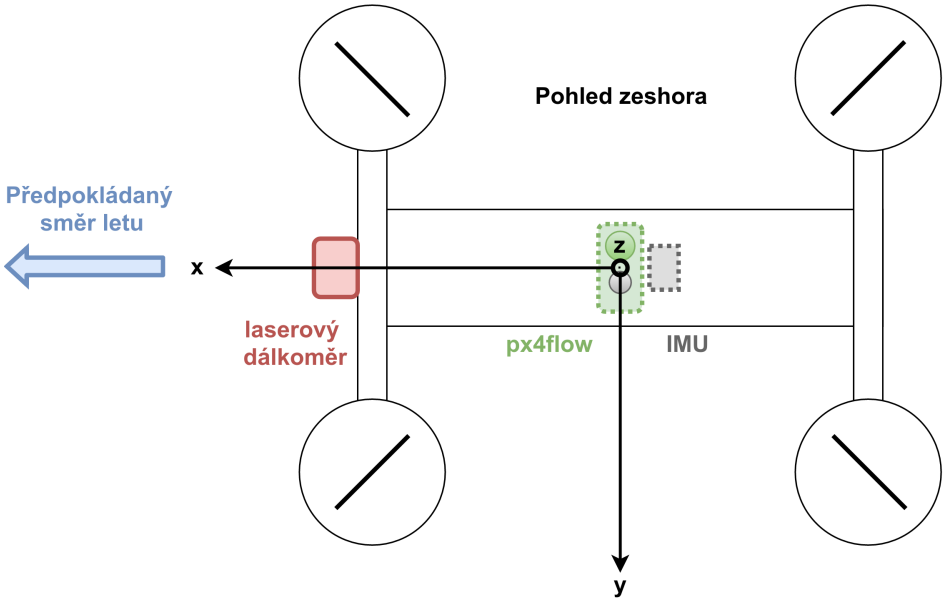
Obrázek 5.5.: Zapojení převodníku logických úrovní do portu GIPO počítače Jetson TK1. Čísla použitých pinů konektoru J3A2: VCC 1.8V - 37, GND - 62, UART_RX - 65, UART_TX - 68

Vlnová délka laserové diody	λ = 785 nm (červená)
Napětí	5 DCV, ± 5%
Odběr proudu	maximálně 500 mA (při startu maximálně 800 mA)
Měřící rozsah	20 mm − 5600 mm
Přesnost	20 mm − 1000 mm : ± 10 mm
	1000 mm − 5600 mm : ± 1% z měření
Rozlišení	vzdálenost: 1 mm
	úhel: 0.36°
Maximální měřený úhel	240°
Trvání pořízení jednoho scanu	100 ms
Maximální snímací frekvence	10 Hz (maximálně 10 scanů/s)
Hmotnost	160 g
Rozměry	50 × 50 × 70 mm
Rozhraní	USB(až 12Mbit/s), RS-232c(až 750kbit/s)

Tabulka 5.1.1.: Nejdůležitější parametry senzoru Hokuyo URG-04LX[2].

5.3. Výběr metody mapování

Výběr vhodné metody jsme prováděli s ohledem na realizovatelnost na dostupném hardwarovém vybavení a plánované použití ve vnitřním prostředí. Z celé řady známých algoritmů jsme nakonec vybrali scan-matching metodu ICP SLAM. Kromě toho, že z podstaty scan-matching algoritmu odpadá potřeba hledat příznaky a jejich korespondující dvojice, algoritmus ICP používají úspěšně autoři [24]. Ve své práci uvádějí jako velkou výhodu nízké nároky na výpočetní výkon a tedy i značnou rychlost algoritmu (autoři používají pro implementaci podobně výkonný počítač, jako je k dispozici v této práci). Dalším rozhodujícím faktorem byla dostupnost kvalitně implementovaného algoritmu ve svobodné knihovně [4].



Obrázek 5.6.: Umístění senzorů na kvadrokoptéře. Osa z směřuje kolmo vzhůru, modul Px4flow a IMU jsou připevněny na spodní části trupu.

5.3.1. ICP algoritmus

Iterative Closest Point (ICP) je scan-matching algoritmus (viz část 4.1.3), který hledá optimální rotačně-translační transformaci mezi dvěmi posloupnostmi bodů (bodovými shluky).

Nechť rotace souřadné soustavy v rovině okolo počátku o úhel φ (v pravotočivé soustavě) je popsána tzv. *rotační maticí*

$$\mathbf{R}_\varphi = \begin{bmatrix} cos(\varphi) & -sin(\varphi) \\ sin(\varphi) & cos(\varphi) \end{bmatrix} \tag{5.3.1}$$

a translaci souřadné soustavy nechť popisuje tzv. *translační matice*

$$\mathbf{t} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} \tag{5.3.2}$$

Dále budeme uvažovat dvě souřadné soustavy, „původní soustavu“ S a „novou soustavu“ S'. Souřadnice bodů v těchto souřadných soustavách označíme popořadě p_i, p'_i. Pak vztah pro transformaci souřadnic bodů ze staré soustavy do nové je dán vztahem

$$p'_i = \mathbf{R}_\varphi p_i + \mathbf{t} \tag{5.3.3}$$

Dále definujeme *tzv. zobecnělou vzdálenost* $E_{dist}(\mathbf{R}_\varphi, \mathbf{t})$, danou vztahem 5.3.4 [31].

$$E_{dist}(\mathbf{R}_\varphi, \mathbf{t}) = \sum_{i=1}^{n} |\mathbf{R}_\varphi \mathbf{p_i} + \mathbf{t} - \mathbf{p_i}'|^2 \tag{5.3.4}$$

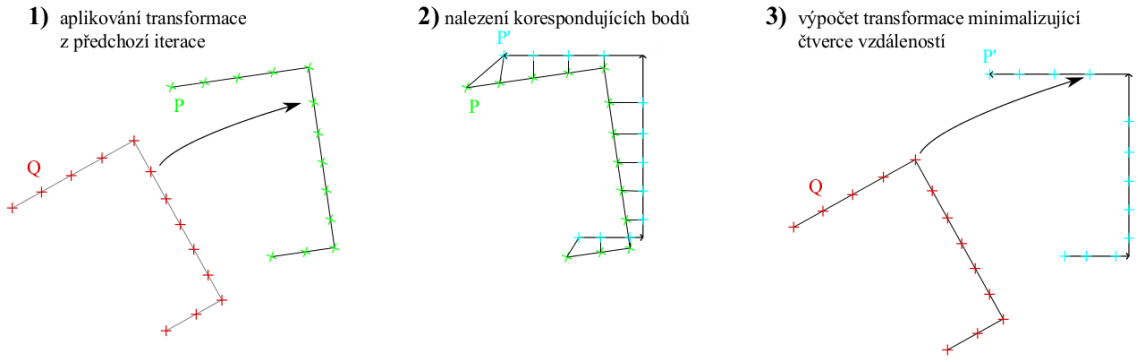
kde $\mathbf{p}_i$, $\mathbf{p}'_i$ matice souřadnicemi bodů dvou bodových posloupností a n je počet bodů v posloupnosti $\mathbf{p}_i$, $\mathbf{p}'_i$.

Za optimální rotačně-translační transformaci, považujeme takovou transformaci, která zobecněnou vzdálenost $E_{dist}(\mathbf{R}_\omega, \mathbf{t})$ minimalizuje. Zobecněnou vzálenost tedy budeme považovat za penalizační funkci, která určuje odlišnost dvou posloupností bodů.

ICP algoritmus pracuje tak, že iterativně zlepšuje nalezenou transformaci mezi dvěmi posloupnostmi bodů. Autor [8] shrnuje algoritmus do tří kroků:

1. Na body z posloupnosti Q (body ze druhé posloupnosti) je aplikována transformace nalezená v předchozí iteraci algoritmu. Pokud se jedná o první iteraci algoritmu, lze ponechat body nezměněné, nebo lze body transformovat na základě odhadu z odometrie robotu. Aplikací transformace na posloupnost Q je získána posloupnost bodů $\mathcal{P}$.
2. Jsou nalezeny korespondující dvojice bodů mezi dvěmi posloupnostmi $\mathcal{P}$ a $\mathcal{P}'$ (první posloupnost). Za dvojici korespondujících bodů zpravidla uvažuje body s nejmenší Euklidovskou vzdáleností.
3. Nové hodnoty transformace x_t, y_t, φ (pro aplikování v dalším kroku iterace) nalezneme minimalizováním $E_{dist}(\mathbf{R}_\varphi, \mathbf{t})$ pro bodové posloupnosti $\mathcal{P}$, $\mathcal{P}$'.

Tyto tři kroky názorně ukazuje obrázek 5.7. Iterace jsou opakovány až do chvíle, kdy rozdíl určených parametrů transformace x_t, y_t, φ klesne pod předem definované konstanty, případně pokud byl překročen maximální definovaný počet iterací. Jak uvádí ve své práci autor [22], pokud jsou dvě posloupnosti bodů $\mathcal{P}'$, Q vzájemně dostatečně „blízko“, lze dokázat, že ICP algoritmus konverguje ke správnému řešení.



Obrázek 5.7.: Jedna iterace ICP algoritmu, převzato z [8], upraveno. Bodové posloupnosti jsou proložené pro přehlednost přímkami.

Výše popsaný algoritmus lze přímo použít pro lokalizaci robotu ve známé mapě, pokud je vždy prováděna registrace příchozího senzorického měření vůdči existující mapě z poslední známé polohy robotu. Malou modifikací, inkrementálním přidáváním nově přijatých senzorických měření do již stávající mapy, získáváme ICP-SLAM algoritmus, který řeší onlineSLAM problém.

Minimalizaci zobecněné vzdálenosti $E_{dist}(\mathbf{R}_\varphi, \mathbf{t})$ lze řešit různými optimalizačními metodami. Klasický přístup zpravidla minimalizaci provádí optimalizační metodou nejmenších čtverců, která má analytické řešení. Vzhledem k tomu, že v této práci používáme knihovnu s již implementovaným algoritmem [4], toto analytické řešení neuvádíme. Celé řešení lze dohledat např. v práci [8].

5.4. Softwarové řešení

V následujících částech popisujeme stěžejní části námi implementovaného programu pro realizaci mapování vnitřního prostředí na autonomní kvadrkokoptéře. Celý program byl implementován v programovacím jazyce C/C++, který byl zvolen především z důvodu kompatibility s existujícími knihovnami a ovladači pro použitý hardware. Pro hlubší pochopení jsou přílohou této práce na CD mimo jiné kompletní zdrojové kódy, ve kterých je (především v příslušných hlavičkových souborech) k dispozici podrobnější dokumentace. Pokud budeme přímo v následujícím textu práce uvádět konkrétní jména funkcí v C++, z důvodu přehlednosti často zanedbáme parametry funkcí či příslušnost ke jmennému prostoru, což lze snadno dohledat v příslušných hlavičkových souborech.

5.4.1. Knihovny MRPT

Při implementaci programu byl využit balík knihoven a aplikací MRPT[2] [4]. Knihovny MRPT jsou napsané v C++ a vyvíjené už od roku 2005. V současnosti obsahují již řadu implementovaných algoritmů používaných nejen v mobilní robotice (SLAM algoritmy, modelování kinematického řetězce, ovladače k senzorickým systémům ap.). Kromě toho jsou k dispozici také podpůrné aplikace, umožňující např. práci s naměřenými daty nebo vygenerovanou mapou a výrazně tak usnadňují vývoj vlastní aplikace. Výhodou také je, že knihovny MRPT nativně podporují práci s laserovým dálkoměrem Hokuyo URG-04LX. Celý balíček knihoven je poskytován volně pod svobodnou licencí BSD. Veškeré software, dostupný na webu [4], je tedy možné nejen zdarma používat, ale i legálně upravovat za podmínky uvedení licence a odkazu na původní autory.

[2] Mobile Robotics Programming Toolkit

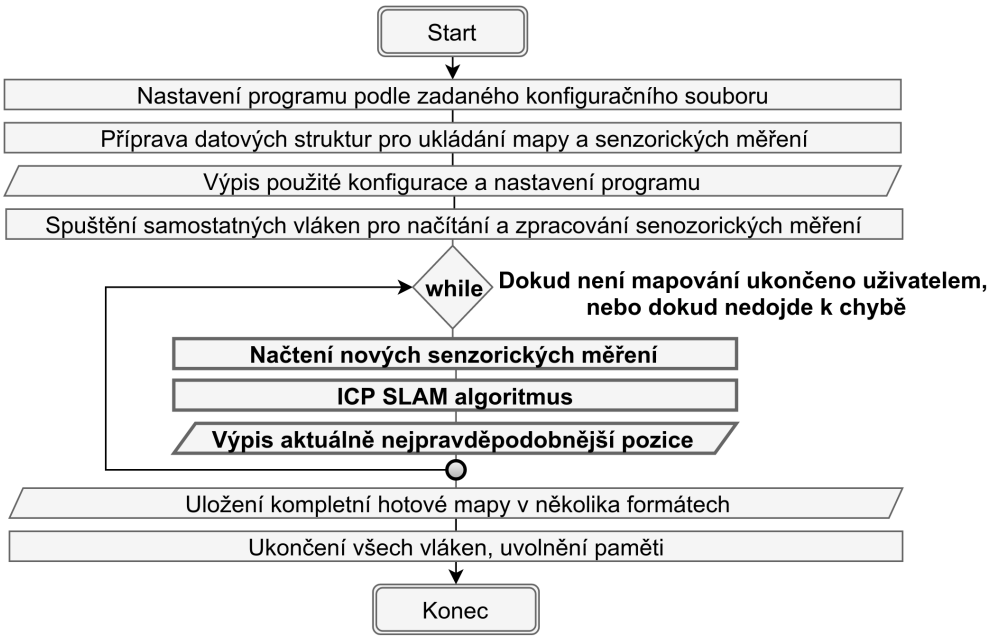
5.4.2. Hlavní funkce programu

Implementovaný program `icp-slam-live` funkčně vychází z ukázkové implementace MRPT knihovny s ICP-SLAM algoritmem, dostupné online [9]. Tato aplikace však byla upravena přímo pro běh v reálném čase na počítači Jetson TK1, tak, aby nebylo nezbytně nutné používat grafické rozhraní (knihovny OpenGL) a byla optimalizována pouze pro běh v konzoli. K původnímu programu, který využíval pouze laserový dálkoměr, byla kompletně implementována videoodometrie založená na datech ze senozorických systémů IMU CHR-6d a modulu Px4flow. Kromě toho byly přidány další funkce jako podpora komunikace počítače Jetson TK1 a Desky regulátorů rychlostí nebo kompenzace náklonů kvadrokoptéry.

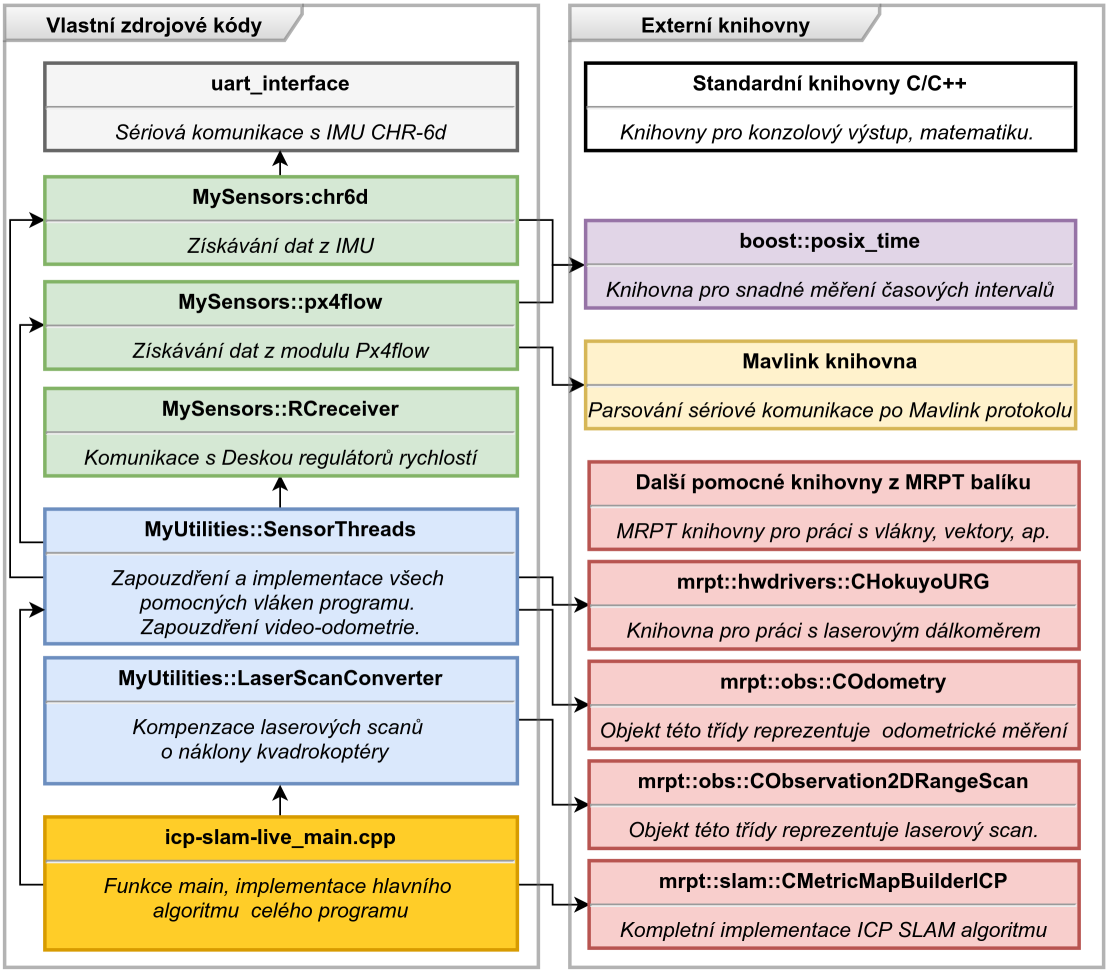
Hlavní funkce programu `icp-slam-live` je řešení fullSLAM problému. Aplikace tedy plní dvě základní funkce. V reálném čase je zároveň:

- inkrementálně budována mapa prostředí
- odhadována okamžitá pozice kvadrokoptéry

Hlavní funkci programu shrnuje vývojový diagram na obrázku 5.8. Zde naznačený cyklus budeme dále označovat jako „hlavní cyklus programu“. Implementace programu se nachází v souboru `icp-slam-live_main.cpp`, pro realizaci dílčích částí programu jsou však používány také další pomocné knihovny, jak je popsáno v následující části 5.4.3.



Obrázek 5.8.: Zjednodušený vývojový diagram aplikace `icp-slam-live`.



Obrázek 5.9.: Členění programu z hlediska tříd a jmenných prostorů.

5.4.3. Členění programu

Při implementaci programu byl kladen důraz na členění zdrojových kódů do tříd a jmenných prostorů dle zásad objektového programování. Přitom bylo v maximální možné míře využito hotových knihoven.

Značně zjednodušené schema návrhu celého programu je uvedeno na obrázku 5.9. Jednotlivé moduly na obrázku vždy označují jednu konkrétní třídu, případně obecněji nějakou knihovnu. Kromě názvu je vždy krátce uveden také její popis hlavní funkce. Barvy odlišují příslušnost k určitému jmennému prostoru či knihovnímu balíku. Šipkami jsou v obrázku naznačeny nejdůležitější vzájemné závislosti mezi jednotlivými moduly.

Všechny námi implementované třídy dodržují konvenci, že jsou vždy implementovány ve dvou souborech, které nesou shodný název jako třída a liší se příponou: *.cpp (zdrojový kód s implementací) a *.h (hlavičkový soubor). Nadále se tedy při popisu implementace nebudeme zabývat názvy souborů. Detailní výpis adresářové struktury je připojen jako příloha.

5.4.4. Vyčítání senzorických dat

Práce s laserovým dálkoměrem Hokuyo URG-04LX, je komfortní, protože existuje již hotová odpovídající knihovna MRPT (`mrpt::hwdrivers::CHokuyoURG`). Způsob práce s touto knihovnou je uveden ve výpisu kódu 5.1. Pořízené laserové scany jsou ukládány do hash mapy laserových scanů, ve které jsou řazeny podle času pořízení. V každé iteraci hlavního programu je díky tomu vyčten pouze nejaktuálnější dostupný scan. Vyčítání je realizováno v samostaném vlákně a pro synchronizaci přístupu k hash mapě více vlákny je použita kritická sekce.

Algoritmus 5.1 Vyčítání a ukládání měření z laserového dálkoměru. Kód je implementován ve funkci `void SensorThreads::SensorThread_laser(TThreadParams params))`

```
#include <mrpt/hwdrivers/CHokuyoURG.h>
using namespace mrpt::hwdriwers;
...
//Spusteni a nastaveni laseroveho dalkomeru:
CGenericSensorPtr sensor = CGenericSensor::createSensorPtr(CHokuyoURG);
sensor->loadConfig(*params.cfgFile,params.section_name);
sensor->initialize();
...
while(!allThreadsMustExit){ //hlavni cyklus vlakna
 sensor->doProcess();
 CGenericSensor::TListObservations lstObjs;
 { //ulozeni mereni do hash mapy v kriticke sekci:
   sensor->getObservations(lstObjs);
   mrpt::synch::CCriticalSectionLocker lock(&cs_global_list_obs);
   global_list_obs.insert(lstObjs.begin(), lstObjs.end());
 }
...
```

IMU CHR-6d sice komunikuje přes standardní sériové rozhraní UART, ale používá svůj vlastní specifický protokol pro přenos informací. Vzhledem k tomu byla podle dodávané dokumentace vytvořena jednoduchá knihovna `uart_interface`, která implementuje základní funkce pro sériovou komunikaci (vyslání a příjem zprávy s určitým obsahem a ID) po použitém protokolu. Implementace parsování je relativně složitá, a proto ji zde neuvádíme. Knihovna byla původně vytvářna se záměřem přímého použití na mikrokontroléru, proto byla napsána pouze v jazyce C, s důrazem na minimální paměťové nároky. Protože se později ukázalo jako výhodnější použít pro začlenění IMU do této práce UART/USB převodník, byla tato knihovna pouze obalena třídou `MySensors::chr6d` v C++ tak, aby šlo formálně i s IMU pracovat stejně jako s dalšími senzory (vytvářet objekt senzoru a volat nad ním funkce). Knihovna při spuštění nastavuje IMU tak, aby IMU automaticky vysílala všechny měřené údaje s maximální možnou frekvencí.

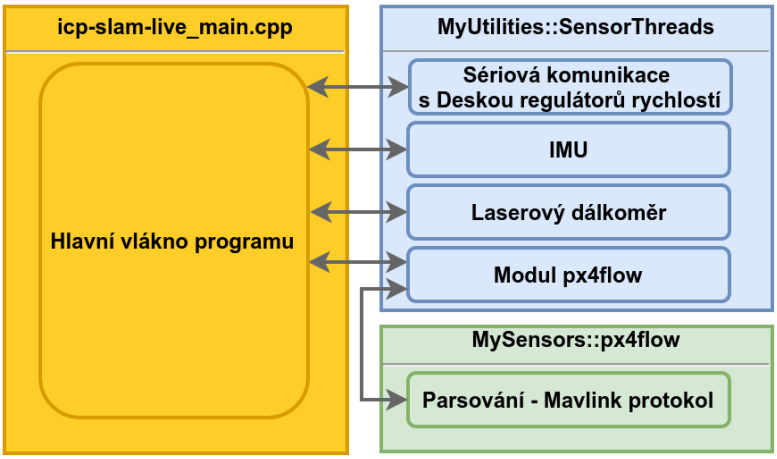
Pro modul px4flow existují sice již hotové knihovny, nicméně tyto knihovny většinou závisejí na celé řadě dalších knihoven a pro účely této práce by bylo jejich použití zbytečně komplikované. Ve třídě `MySensors::px4flow` byla proto vytvořena vlastní knihovna, která používá pouze

originální knihovny Mavlink pro parsování příchozích zpráv a jinak nevyžaduje běh dalších subsystémů (např. běh systému ROS[3]). Způsob parsování příchozí komunikace knihovnou Mavlink je patrný z výpisu kódu 5.2.

Algoritmus 5.2 Parsování zpráv z modulu Px4flow po Mavlink protokolu. Implementováno ve funkci `void *MySensors::px4flow::px4flowSensorThread()`

```
#include <pixhawk/mavlink.h>
#include <mavlink_helpers.h>
...
int bytesTransferred = (int) read(fd, m_buffer, 1);

mavlink_message_t message;
mavlink_status_t status;
for (size_t i = 0; i < bytesTransferred; i++) {
   bool msgReceived = mavlink_parse_char(MAVLINK_COMM_1,
m_buffer[i], &message, &status);
   if (msgReceived) {
     switch (message.msgid) {
      case MAVLINK_MSG_ID_OPTICAL_FLOW: //ID = 100
        {mavlink_optical_flow_t flow;
         mavlink_msg_optical_flow_decode(&message, &flow);
         //ziskani rychlosti ve smeru osy x v m/s:
         rawData.velocity_x = flow.flow_comp_m_x;
         ...
```



Obrázek 5.10.: Struktura vláken a jejich vzájemná komunikace.

Pro možnost současného zpracovávání senzorických dat a běhu hlavního programu byl program implementován ve více vláknech (viz obrázek 5.10). Z hlavního vlákna, ve kterém běžní hlavní cyklus programu, jsou vytvářena i ukončována všechna dílčí vlákna.

[3]Z anglického Robotic Operating System.

5.4.5. Implementace videoodometrie

Předpokládáme pohyb kvadrkoptéry pouze v horizontální rovině. Z modulu Px4flow získáváme odhady okamžité rychlosti $\vec{v}$ po složkách ve dvou horizontálních osách v_x, v_y. Pro polohu $\vec{s} = (x, y)$ a rychlost $\vec{v} = (v_x, v_y)$ platí známý vztah [7]:

$$\vec{s} = \vec{s_0} + \int \vec{v}\,(t)\ dt \tag{5.4.1}$$

kde $\vec{s_0} = (x_0, y_0)$ je počáteční poloha a t je doba pohybu. V tomto jednoduchém případě (za předpokladu konstantní rychlosti $\vec{v}$) můžeme souřadnice aktuální pozice x, y určit jako:

$$x = x_0 + v_x t, \qquad y = y_0 + v_y t \tag{5.4.2}$$

Pokud tedy měříme čas mezi příchozími měřeními z modulu Px4flow, můžeme iterativně odhadovat aktuální souřadnice kvadrokoptéry vzhledem k počátku podle vztahů 5.4.2, s tím, že za počáteční polohu vždy dosadíme předchozí odhadnutou pozici.

Analogickým postupem lze odhadnout aktuální hodnotu úhlu yaw (natočení kolem osy z), pouze je integrována místo dopředné úhlová rychlost z IMU.

Pro ukládání videoodometrie byla použita knihovna `mrpt::obs::CObdometry`. Aby bylo odometrické měření správně zpracováno dalšími knihovnami z balíku MRPT, je nezbytné vždy ukádat absolutní naintegrovanou pozici od začátku mapování. Integrace rychlostí a ukládání polohy je realizováno ve třídě `MyUtilities::SensorThreads`. V hlavním programu se tak již nepracuje se surovými daty ze senzorů, ale přímo s instancemi třídy `CObdometry`. Odometrické měření je získáváno na začátku každé iterace hlavního cyklu programu funkcí `getVideoOdometry` z knihovny `MyUtilities::SensorThreads`. Pro měření časových úseků byla použita knihovna `boos::posix_time`.

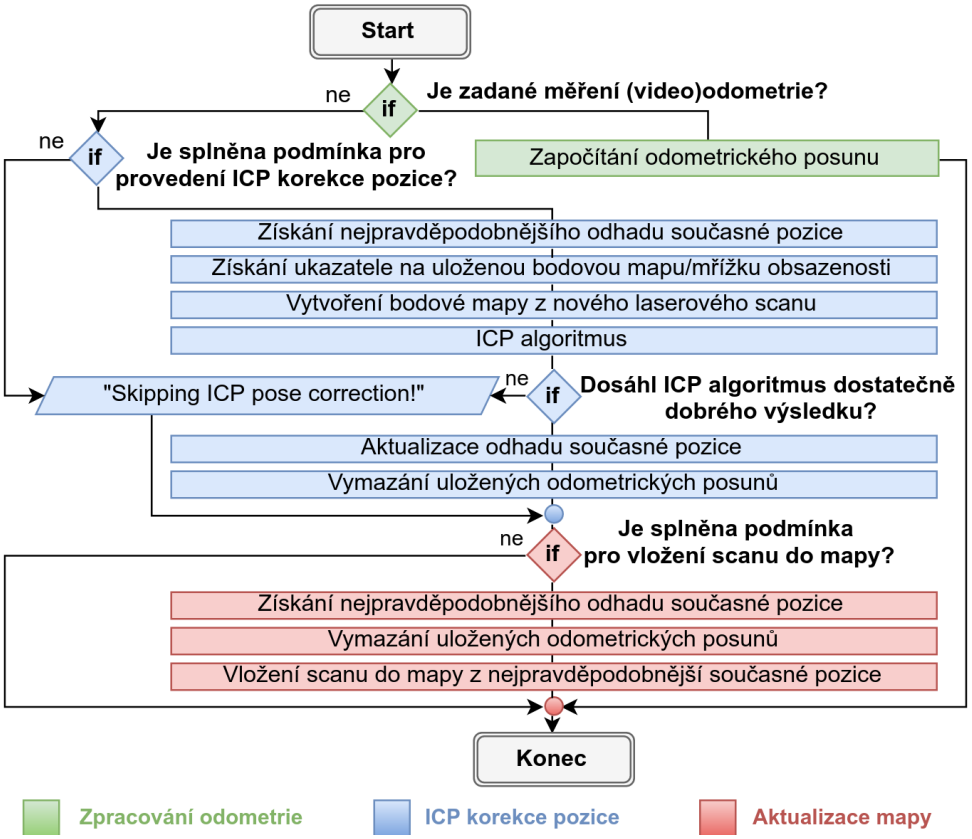
5.4.6. Implementace ICP SLAM algoritmu v MRPT knihovně

ICP-SLAM algoritmus je implementován v MRPT knihovně `CMetricMapBuilderICP`. Klíčovou funkcí knihovny je funkce `processObservation`, která provede zahrnutí laserového scanu či odometrie do ICP-SLAM algoritmu (viz obrázek 5.11).

Klíčové pro chování celého ICP-SLAM algoritmu ve funkci `processObservation` jsou podmínky, které rozhodují o vykonání posledních dvou funkčních bloků (provedení ICP korekce pozice, aktualizace mapy).

ICP korekce pozice je vynechána, pokud:

- bylo zaznamenáno nějaké odometrické měření od doby poslední aktualizace pozice ICP algoritmem



Obrázek 5.11.: Zjednodušený vývojový diagram funkce `void processObservation(mrpt::obs::CObservation &obs)` z knihovny `mrpt::slam::CMetricMapBuilderICP`

- a zároveň je změna vzdálenosti a úhelu od poslední ICP korekce polohy menší než definované limity `localizationLinDistance`, `localizationAngDistance`

Pokud je podmínka nesplněna, pak se provede ICP algoritmus tak, jak byl popsán v části 5.3.1. Jako počáteční odhad pozice se pro ICP algoritmus použije nejaktuálnější dostupný odhad polohy kvadrokoptéry S, který je získán jako

$$S = S_{ICP} + \left(S_{odo} - S_{ref}\right) + extrapolace\ z\ rychlostí \tag{5.4.3}$$

kde S_{ICP} je poslední poloha potvrzená ICP algoritmem a rozdíl $(S_{odo} - S_{ref})$ představuje odometrický posun od poslední ICP korekce polohy. Pro uchování informace o pozici robotu je použita třída `CRobot2DPoseEstimator`, která mimo jiné provádí také lineární extrapolaci z uložených rychlostí, které jsou průběžně počítány při každém vložení nového senzorického měření.

Aktualizace mapy proběhne, pokud:

- jde o první provedené senzorické měření

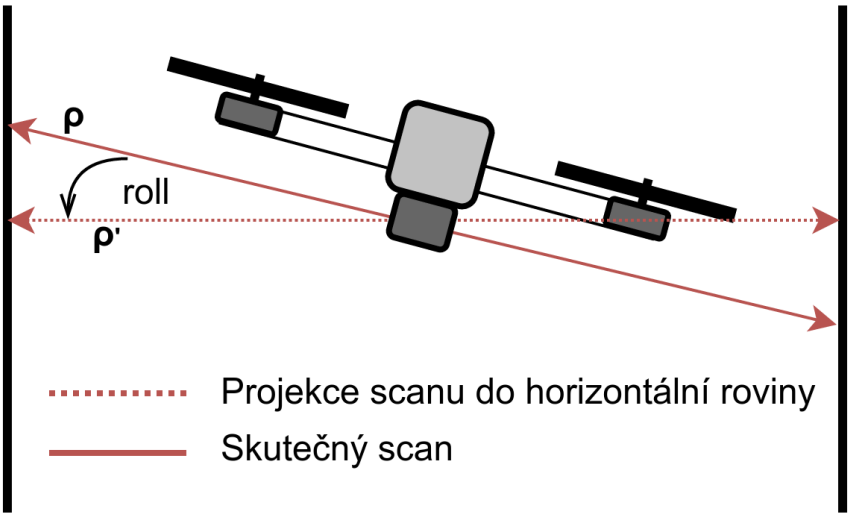
- nebo bylo při ICP korekci dosaženo dostatečně dobrého výsledku (limit `minICPgoodnessToAccept`)
- a zároveň je změna vzdálenosti a úhlu od posledního vložení do mapy větší nebo rovna definovaným limitům `insertionLinDistance`, `insertionAngDistance`

Samotné vložení pozorování do mapy (mřížky obsazenosti) je provedeno dalšími pomocnými knihovnami MRPT podobně, jako bylo zmíněno v kapitole 2 (použitím Bayesovy věty).

Chování ICP-SLAM algoritmu ovlivňuje také nastavení samotného ICP algoritmu. Možnosti nastavení ICP algoritmu jsou uvedeny v části 6.3, případně v příloze B.

5.4.7. Kompenzace náklonů helikoptéry

Data z laserového dálkoměru na kvadrokoptéře jsou narozdíl od aplikací na kolovém robotu zatížena chybou vznikající při náklonech kvarokoptéry. Vzhledem k tomu, že námi použitá kvadrokoptéra nedisponuje žádnou stabilizovanou platformou, které by změny orientace kvadrokoptéry potlačovala a implementace ICP-SLAM algoritmu předpokládá stabilní pozici laserového dálkoměru, navrhli jsme jednoduchou kompenzaci, která se pokouší opravovat laserové scany o náklony kvadrokoptéry roll a pitch. Protože jsme přijali předpoklad, že zdi i stěny překážek jsou kolmé vzhledem k podlaze vnitřního prostředí (horizontální rovině), můžeme dopočítat ze znalosti úhlů roll a pitch teoretický průmět laserového scanu do horizontální roviny. Princip průmětu je patrný z obrázku 5.12. Analogickým způsobem můžeme uvažovat průmět pro úhel pitch.



Obrázek 5.12.: Průmět laserového scanu do horizontální roviny. Pohled zepředu, kvadrokoptéra je natočena o úhel roll.

Předpokládáme, že orientace laserového dálkoměru během snímání jednoho scanu je konstantní (jedno sejmutí laserového scanu trvá pouze 0.1 s). Nechť $\vec{\rho}$ je vektor naměřených vzdáleností (jeden laserový scan), získaný při orientaci kvadrokoptéry: roll = α, pitch=β. Z jednoduchého

geometrického náhledu je zřejmé, že průmět laserového scanu do horizontální roviny $\vec{\rho}'$ je dán jako:

$$\vec{\rho}' = \vec{\rho} \cdot |cos(\alpha) \cdot cos(\beta)| \tag{5.4.4}$$

Absolutní hodnota ve vztahu 5.4.4 sice není z matematického hlediska nutná, nicméně zaručuje, že vztah bude univerzálně platit pro všechny kvadranty (vzdálenost k překážce je vždy kladné číslo).

Takto byl filtr implemetován ve třídě `MyUtilities::LaserScanConvertor`. Kromě toho jsme definovali konstanty, kterými lze nastavit limitní hodnoty úhlů a výšky kvadrokoptéry. Lze tak určit, pro jaké hodnoty úhlů se korekce ještě provede a pro jaké hodnoty úhlů již není laserový scan brán v potaz. Více informací o nastavení filtru je uvedeno v příloze B.

5.4.8. Ukládání mapy

Pro ukládání mapy používá knihovna `CMetricMapBuilderICP` několik různých formátů (pomocných knihoven MRPT).

Základním úložištěm pro naměřená senzorická data je třída `CSimpleMap`. Třída `CSimpleMap` uchovává v paměti mapu jako spojový seznam, tvořený dvojicemi „poloha pořízení, získaná senzorická měření“. Uložené pozice nejsou ukládány absolutně, ale jako distribuce rozložení pravděpodobnosti skutečné polohy robotu. Z takto uložených senzorických dat lze získat zpětně další typy map, např. mřížku obsazenosti. Třída navíc poskytuje metody pro uložení komprimované mapy do binárního souboru.

Druhou používanou skupinou map jsou tzv. metrické mapy, které mají v MRPT společného předka - virtuální třídu `CMetricMap`. Do této skupiny spadají mapy jako např. mřížka obsazenosti, prostá bodová mapa, příznaková mapa ap. Tyto mapy lze jednoduše převést do grafické podoby srozumitelné člověku.

Ve funkci `processObservation` (viz obrázek 5.11, blok aktualizace mapy) je vždy aktualizována mapa ve formátu `CSimpleMap` a kromě toho podle použitého nastavení buď mřížka obsazenosti, nebo prostá bodová mapa. Zatímco mapa ve formátu `CSimpleMap` slouží především pro úsporné uložení zaznamenaných dat pro uživatele, mřížky obsazenosti či prosté bodové mapy jsou používány přímo při korekci polohy ICP algoritmem jako reference pro nově příchozí laserový scan.

Jakmile je ukončen cyklus hlavního programu, je provedeno automatické uložení celé získané mapy do souboru s příponou .simplemap. Kromě toho je volitelně také vyexportována přímo mřížka obsazenosti jako obrázek PNG a dále je ukládána i prostá bodová mapa jako textový soubor se souřadnicemi bodů (souřadnice jednoho bodu jsou na jednom řádku).

5.5. Poznámky k používání programu

5.5.1. Kompilace programu ze zdrojových kódů

Pro sestavení aplikace je používám program CMake, který by měl zaručovat snadnou přenositelnost programu na různé platformy. Pro úspěšnou kompilaci programu je nezbytné splnění následujících závislostí:

- CMake, verze 2.8 či vyšší
- balík knihoven MRPT, verze 1.3.2 či vyšší
- knihovna Boost system, verze 1.60 či vyšší
- standardní knihovny pro vstup a výstup v C++
- knihovny OpenGL (je-li program kompilován s grafickým uživatelským rozhraním, viz část 5.5.3)

Zdrojové kódy knihoven Mavlink byly přidány k této práci a není tedy nutná jejich samostatná instalace.

Makefile je vytvořen příkazem `cmake` v adresáři obsahujícím soubor `CMakeList.txt`. Proběhne-li vytváření korektně, je možné program zkompilovat příkazem `make` ve stejném adresáři.

Program byl úspěšně otestován na architektuře x86 (klasické PC - Lenovo ThinkPad X201, OS Linux Mint 17) i ARM (NVIDIA Jetson TK1, OS Ubuntu).

5.5.2. Start aplikace

Program je koncipován pro spouštění z konzole standardním způsobem, jako jediný parametr programu je předávána cesta ke konfiguračnímu souboru. Program spustíme zadáním[4]:
`./icp-slam-live config.ini`

Pokud není cesta ke konfiguračnímu souboru uvedena, skončí aplikace výjimkou a uživateli je zobrazena nápověda se správnou syntaxí. Pro korektní spuštění programu je také nezbytné nejprve správně nastavit parametry programu v konfiguračním souboru (viz část 5.5.4).

5.5.3. Grafické uživatelské rozhraní

Grafické rozhraní (GUI) původní ukázkové aplikace `icp-slam-live` je velmi praktické pro účely ladění programu (zobrazuje v reálném čase pozici robotu, vybudovanou mapu i aktuální dosah laserového dálkoměru), ale nehodí se přímo pro použití na kvadrokoptéře. Proto bylo nakonec zachováno, ale tak aby bylo možné jeho použití snadno vypnout. Vypnutí/zapnutí GUI lze provést před kompilací na začátku zdrojového kódu v souboru `icp-slam-live_main.cpp` změnou konstanty `USE_GUI` na hodnotu 0 či 1.

[4] Tento příklad syntaxe předpokládá přítomnost konfiguračního souboru „`config.ini`“ ve stejném adresáři, jako se nachází zkompilovaný program.

5.5.4. Možnosti nastavení

Veškerá nastavení aplikace (kromě případné aktivace GUI) se provádějí editací parametrů v konfiguračním .ini souboru.

Struktura .ini souboru byla zachována z ukázkového programu a význam parametrů tak lze dohledat na webu autorů [4], okomentovaný ukázkový konfigurační soubor včetně všech přidaných parametrů pro popisovanou kvadrokoptéru je připojen v příloze B.

Zde popíšeme pouze strukturu konfiguračního souboru a zdůrazníme některé klíčové parametry, které je nutné nastavit před spuštěním programu, nebo které mají významný vliv na funkci programu.

Nastavení senzorů se provádí v sekcích: `[LASER]`, `[PX4FLOW]`, `[CHR6D]`, `[RCreceiver]`. Velmi důležité je před spuštěním programu ověřit, zda odpovídají skutečnosti sériové porty (parametr `COM_port_LIN`), na kterých jsou v systému senzory připojeny. Aplikace předpokládá pouze vyplnění názvu zařízení, tak jak je uvedeno ve standardním adresáři `/dev`. Parametr `driver` umožňuje nastavit použitý ovladač pro laserový dálkoměr z knihovny MRPT, což dovoluje velmi snadnou záměnu laserového dálkoměru za jiný. Důležité je správně nastavit orientaci laserového dálkoměru podle jeho fyzického uchycení na kvadrokoptéře.

Videoodometrii lze povolit parametrem `useOdometry` v sekci `[VideoOdometry]`. Navíc lze zvlášť rozhodnout, zda se má do odometrie používat také odhad úhlu yaw integrací úhlových rychlostí z IMU (parametr `estimateYawDisplacement`). Pokud je tato možnost zakázána, je při úhlových změnách plně spoléháno pouze na laserová měření a ICP algoritmus.

V sekci `[LaserScanConverter]` je možné nastavit parametry kompenzování náklonů kvadrokoptéry (viz část 5.4.7). Používání kompenzace se povoluje parametrem `useLaserScanConverter`.

Sekce `[ICP]` a `[MappingApplication]` obsahují parametry, které přímo ovlivňují chování ICP-SLAM algoritmu. Hledání vhodných hodnot těchto parametrů se věnuje část popisující experimenty 6.3.1. V sekci `[MappingApplication]` lze dále najít parametry ovlivňující chování aplikace `icp-slam-live` jako celku. Kromě možnosti povolit ukládání záznamu všech surových senzorických dat (tzv. „raw logů“) jsou zde také parametry pro nastavení GUI.

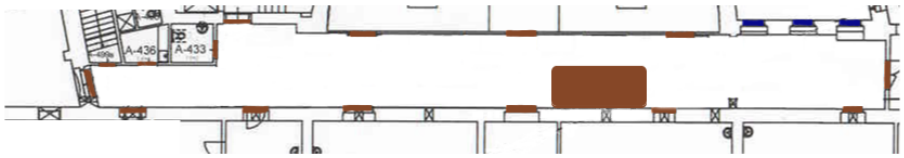
V posledních sekcích je pak možnost nastavit parametry generovaných výstupních map (např. rozlišení mřížky obsazenosti ap).

Kapitola 6.

6 Experimentální výsledky

V této kapitole nejprve rozebereme experiment s původní ukázkovou implementací ICP-SLAM algoritmu v MRPT knihovně `CMetricMapBuilderICP`. Na základě nedostatků této implementace bylo navrženo řešení pro mapování vnitřního prostředí kvadrokoptérou, popisované v kapitole 5. Ve zbytku této kapitoly uvedeme výsledky námi navrženého mapování na kterých diskutujeme nalezení vhodných parametrů. Získané výsledky porovnáváme s referenční technickou dokumentací.

6.1. Původní aplikace icp-slam-live

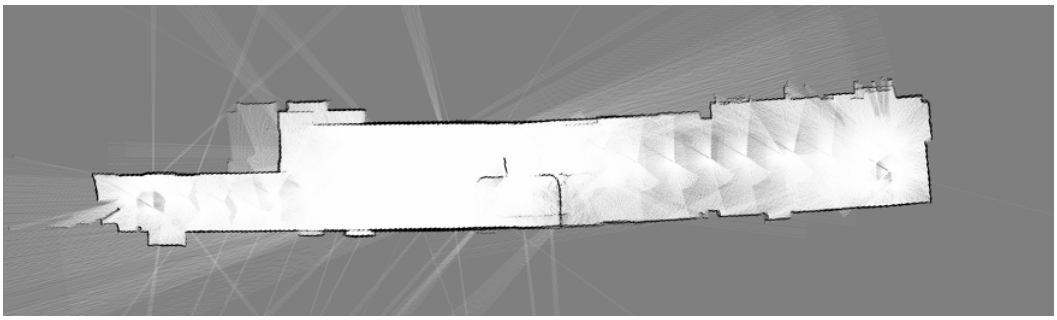


Obrázek 6.1.: Technická dokomentace mapovaného prostoru, upravená pro potřeby porovnání se získanou mapou.

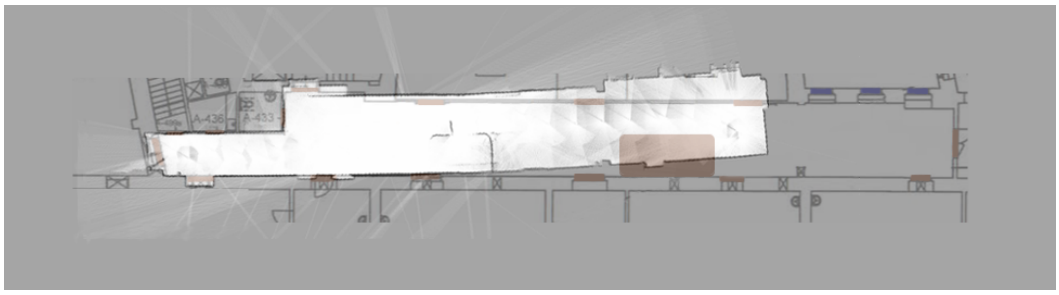
Pro prvotní ověření funkce knihovny `mrpt::slam::CMetricMapBuilder` byla použita ukázková implementace v aplikaci `icp-slam-live` [9], která používá pro svoji činnost pouze laserový dálkoměr. Tento experiment probíhal v prostorách budovy ČVUT na Karlově náměstí, kde byla mapována chodba, která kromě členitějšího prostředí obsahuje také dlouhý rovný úsek. K mapovaným prostorům byla získána technická dokumentace (viz obrázek 6.1), kterou budeme uvažovat jako referenci pro vytvářenou mapu. Do technické dokumentace bylo dokresleno stanoviště pro dozor učeben, které v původních plánech budovy nebylo zahrnuto. Mapování probíhalo

od levého konce směrem doprava a zpět. Pro experiment jsme použili ukázkové výchozí nastavení [9].

S laserovým dálkoměrem Hokuyo URG-04LX byl prováděn pohyb simulující let kvadrokoptéry (nestabilní rychlost pohybu, změna výšky v rozmezí cca 1m, mírné náklony laserového dálkoměru).



Obrázek 6.2.: Získaná mapa prostředí.



Obrázek 6.3.: Vzájemný proklad získané mapy a skutečného výkresu chodby ve stejném měřítku.

Z obrázku 6.3 je zřejmé, že vytvořená mapa je nepřesná a pouze vzdáleně tvarem odpovídá skutečnosti. Mapa chodby má především výrazně kratší délku a vlivem toho bylo stanoviště pro službu umístěno o několik metrů jinde, než je jeho skutečná pozice. Zatímco z počátku (levá strana chodby) mapa relativně věrně kopíruje skutečnost, na druhém konci chodby již dochází k velkým odchylkám a je patrná narůstající chyba.

Ze vzájemného srovnání na obrázku 6.3 lze vypozorovat, že mapa byla zkrácena především tam, kde je prostředí skoupé na výraznější nerovnosti, zatímco šíře chodby byla po celou dobu modelována relativně přesně.

Selhání ICP-SLAM algoritmu zde lze vysvětlit omezeným dosahem použitého laserového dálkoměru. Výrobce udává maximální dosah dálkoměru zhruba 5.6 m (viz tabulka 5.1.1), zatímco mapovaná chodba je dlouhá řádově desítky metrů. Protože laserový dálkoměr většinu mapované trasy „nevidí“ konec chodby, ICP-SLAM algoritmus spoléhá pouze na vzdálenosti ke stěnám chodby po stranách.

Pokud tedy nastane situace, že je mapován delší úsek chodby, která z hlediska získávaných dat nevykazuje žádené výraznější rozdíly, ICP-SLAM algoritmus není schopen rozpoznat korektně pohyb robotu a vytvořit odpovídající mapu.

6.2. Videoodometrie

Implementace videoodometrie byla nejprve samostatně otestována v rozdílných prostředích, ve kterých poté probíhalo také testování navrženého řešení mapování. Jak se ukázalo, přesnost odhadu rychlostí (a tedy i odhadované pozice kvadrokoptéry) z modulu Px4flow silně závisí na typu podlahy a intenzitě osvětlení. Vzhledem k faktu, že intenzita osvětlení v mapovaném prostředí kolísá a neměli jsme možnost ji objektivně měřit, uvádíme alespoň odhad přibližné hodnoty chyby videoodometrického měření pro mapovaná prostředí, založený na opakovaném měření známé referenční vzdálenosti při konstantní výšce modulu Px4flow nad podlahou. Odhady shrnuje tabulka 6.2.1.

Typ prostředí	**Obrázek:**	**Podlaha:**	**Osvětlení:**	**Odhad chyby měření vzdálenosti v %:**
Chodby v administrativní budově	6.4a	tmavý koberec	umělé, místy přirozené	30%
Vstup k výtahům	6.4b	broušené teraso	umělé	15%

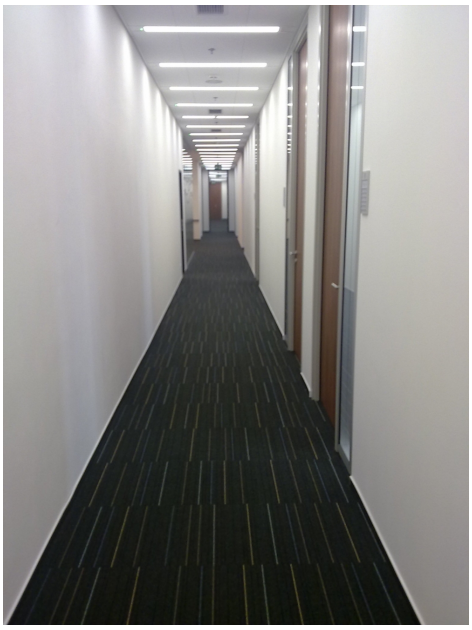
Tabulka 6.2.1.: Závislost videoodometrického měření vzdáleností na okolních podmínkách.

Chyby vzdáleností jsou menší, pokud bylo použito přídavné osvětlení výkonnými reflektory, nicméně stále byl výrazný rozdíl v přesnosti měření v různých prostředích. Kromě toho byla pozorována také určitá závislost na rychlosti pohybu, především v chodbách, kde se vzor koberce neustále opakuje.

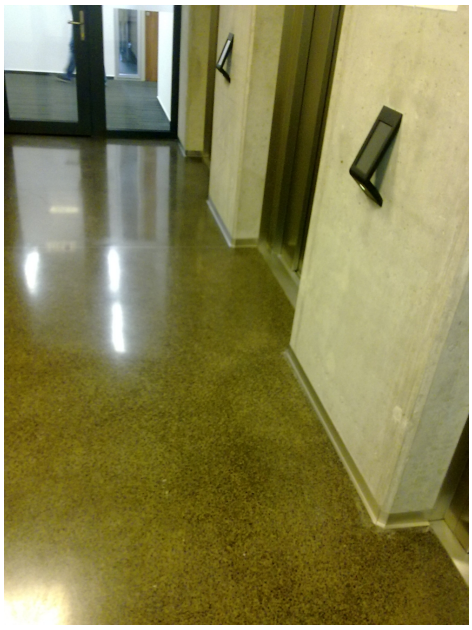
Při testování integrace úhlu se ukázalo, že data z použitého gyroskopu (IMU) jsou velmi silně ovlivněna otřesy helikoptéry a výsledný integrovaný úhel byl zatížen vysokou chybou (po silných otřesech dával gyroskop často i nesmyslné hodnoty). Z tohoto důvodu byla při následujících experimentech použita pouze integrace dopředných rychlostí z modulu Px4flow.

6.3. Experimenty s testovací platformou

Pro testování a ladění navrženého softwarového řešení byla použita improvizovaná testovací platforma (držák na kvadrokoptéru s pojezdem), která umožňuje simulovat běh algoritmu na letící kvadrokoptéře, bez přímé nutnosti letu kvadrokoptéry. Testování algoritmu tak bylo možné uskutečnit i v místech, kde by byl let s kvadrokoptérou problematický. Pro ladění parametrů algoritmu



(a) Chodba administrativní budovy



(b) Vstupní místost k výtahům

Obrázek 6.4.: Mapované typy prostředí

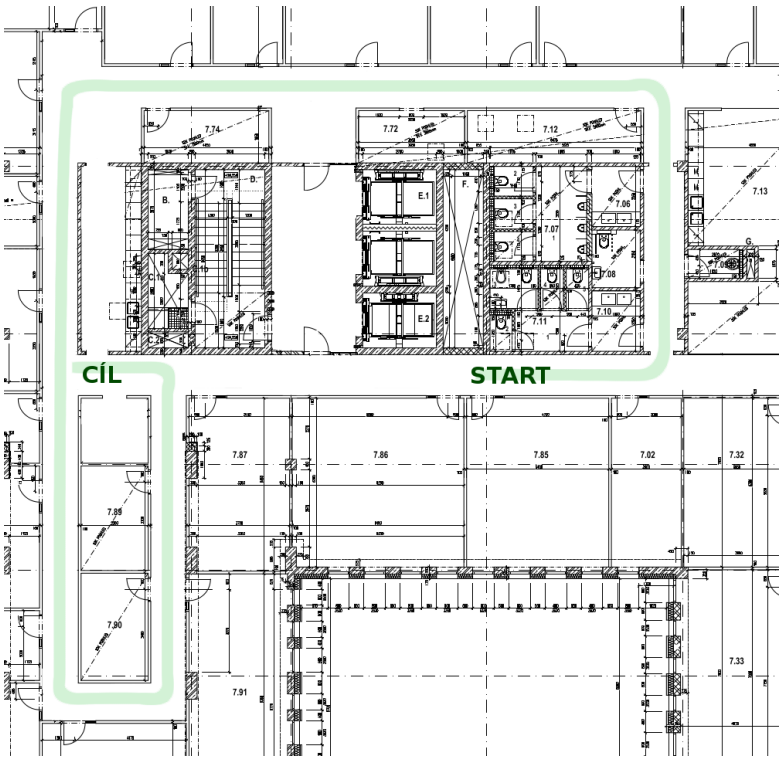
bylo používáno záznamu všech senzorických měření, na kterých byl poté spouštěn totožný ICP-SLAM algoritmus offline.

Jako výchozí nastavení pro experimenty považujeme konfigurační soubor, připojený v příloze B, u získaných map vždy uvedeme parametry, které byly pozměněny. Jako referenci pro vytvořené mapy budeme opět uvažovat upravenou technickou dokumentaci mapované budovy. Získané mapy budeme uvádět vždy ve formátu obrázku exportovného z mřížky obsazenosti. Takto získaná grafická podoba mapy lépe vypovídá o procesu mapování než pouhá bodová mapa.

6.3.1. Dlouhá chodba s uzavřením smyčky

Nejprve byla mapována relativně dlouhá trasa v chodbách (viz obrázek 6.4a). Zvolená trasa (viz obrázek 6.5) umožňuje dobře otestovat integraci videoodometrie do ICP-SLAM algoritmu, bez které mapování založené pouze na laserovém dálkoměru v tomto typu prostředí selhává. Zároveň byla trasa pro mapování zvolena tak, aby došlo k návratu na stejné místo (uzavření smyčky). Pro zjednodušení byl tento experiment prováděn bez simulace výraznějších náklonů kvadrokoptéry (maximální změny úhlů roll a pitch z rovnovážené polohy byly maximálně $\pm 3°$). S ohledem na velikost mapovaného prostředí bylo zvoleno rozlišení mřížky obsazenosti 0.02m. Některé z použitých parametrů spolu s odkazy na korespondující mapy jsou uvedeny v tabulce 6.3.1.

Význam prvních čtyř parametrů v tabulce 6.3.1 byl již vysvětlen v části 5.4.6. Tyto parametry hrají klíčovou roli při rozhodování o provedení ICP korekce pozice a vložení scanu do mapy.

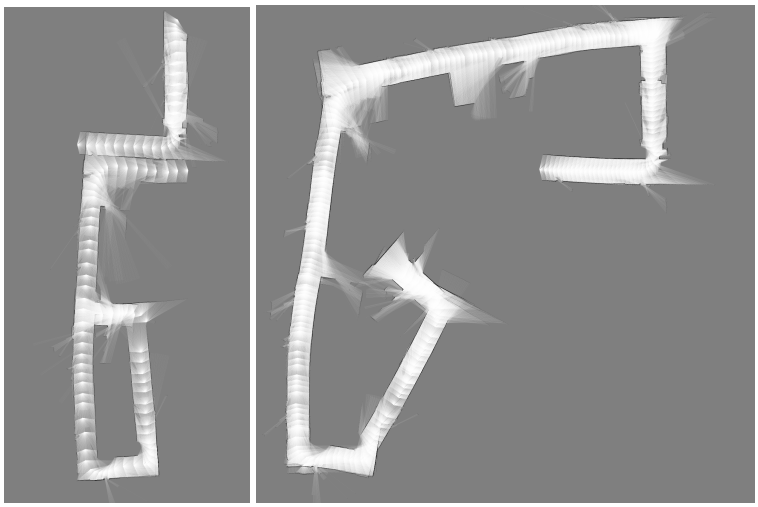


Obrázek 6.5.: Trasa mapování dlouhé chodby.

Pro námi implementované řešení se jeví jako ideální, nastavit hodnoty pro provedení ICP korekce pozice (`localizationLinDistance`, `localizationAngDistance`) na velmi malé hodnoty. Díky tomu je ICP korekce prováděna velmi často. I na rovných úsecích je tak zaznamenáván pohyb kvadrokoptéry. Podobně to platí také o parametrech ovlivňující vkládání scanu do mapy (`insertionLinDistance`, `insertionAngDistance`). Zde však musí být hodnoty celkově vyšší, jinak by docházelo při velmi častém vkládání scanů do mřížky obsazenosti k chybám (pokud se chybně nastaví v mřížce obsazenosti vysoká pravděpodobnost volného prostoru, algoritmus se s touto chybou těžko vypořádá).

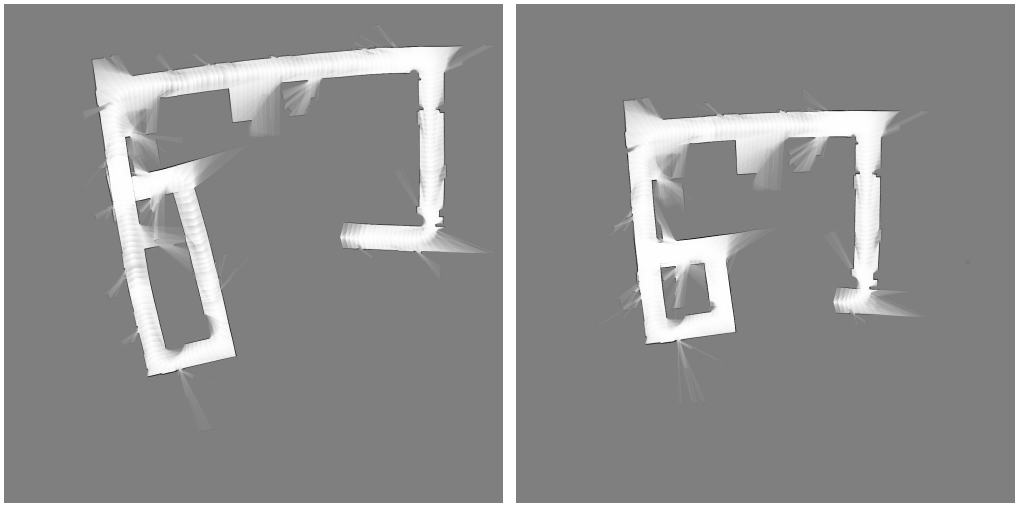
	Číslo měření:	**1**	**2**	**3**	**4**	**5**	**6**
	Obrázek s výslednou mapou	6.6a	6.6b	6.7a	6.7b	6.8a	6.8b
Hodnoty parametrů	`localizationLinDistance` [m]	0.1	0.18	0.02	0.02	0.02	0.02
	`localizationAngDistance` [°]	5	5	2	2	2	2
	`insertionLinDistance` [m]	0.6	0.25	0.25	0.25	0.25	0.25
	`insertionAngDistance` [°]	10	8	8	8	8	8
	`thresholdDist` [m]	0.2	0.2	0.4	0.2	0.2	0.2
	`tresholdAng_DEG` [°]	5	5	10	5	5	5
	`minICPgoodnessToAccept`[%]	40	40	40	40	40	60
	Použití videoodometrie	ANO	ANO	ANO	NE	ANO	ANO

Tabulka 6.3.1.: Mapování dlouhé chodby. Některé použité parametry a výsledné mapy.



(a) (b)

Obrázek 6.6.: Mapování dlouhé chodby - získané mapy



(a) (b)

Obrázek 6.7.: Mapování dlouhé chodby - získané mapy

Pokud je prostředí dostatečně členité, lze hodnoty parametrů pro opravu pozice i vkládání do mapy zvýšit, stále je však potřeba dodržet pravidlo, že hodnoty vzdáleností pro vložení do mapy jsou vždy vyšší než hodnoty, při kterých se provádí ICP korekce pozice (hodnoty úhlů mohou být podobné).

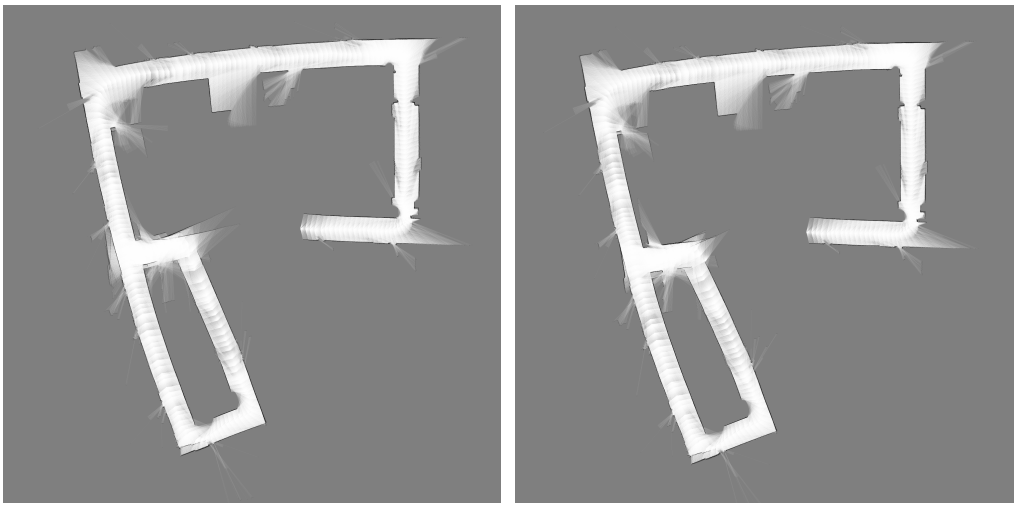
Příliš vysoké hodnoty parametrů byly nastaveny v případě mapy na obrázku 6.6a, zde algoritmus úplně selhává. Algoritmus částečně chybně určuje úhly také u mapy na obrázku 6.6b.

Navyšování počtu iterací ICP algoritmu nepřineslo takřka žádné viditelné zlepšení (data k těmto experimetům nejsou uvedena v tabulce 6.3.1).

Parametry `thresholdDist` a `tresholdAng_DEG` ovlivňují chování samotného ICP algoritmu. Tyto parametry určují počáteční maximální vzdálenost a velikost úhlu, pro které mohou být ještě dva body označeny jako bodový pár.

Ukázalo se, že pro příliš nízké hodnoty (vzdálenost zhruba 0.15m a nižší, úhel 2° a nižší) nebyly korektně nalezeny všechny dvojice bodů a mapování selhávalo. Naopak pro příliš vysoké hodnoty (vzdálenost 0.3m, úhel 15° a vyšší) docházelo k výraznému zkrácení rovných úseků (viz obrázek 6.7a).

Relativně věrná mapa z hlediska délky chodeb je uvedena na obrázku 6.8a. Při zachování stejného nastavení byla dále zvýšena hodnota paramatru `minICPgoodnessToAccept`. Tím byly pro stavbu mapy použity pouze kvalitněji interpretované scany a tako byla získána patrně nejvěrnější mapa chodby (viz obrázek 6.8b).



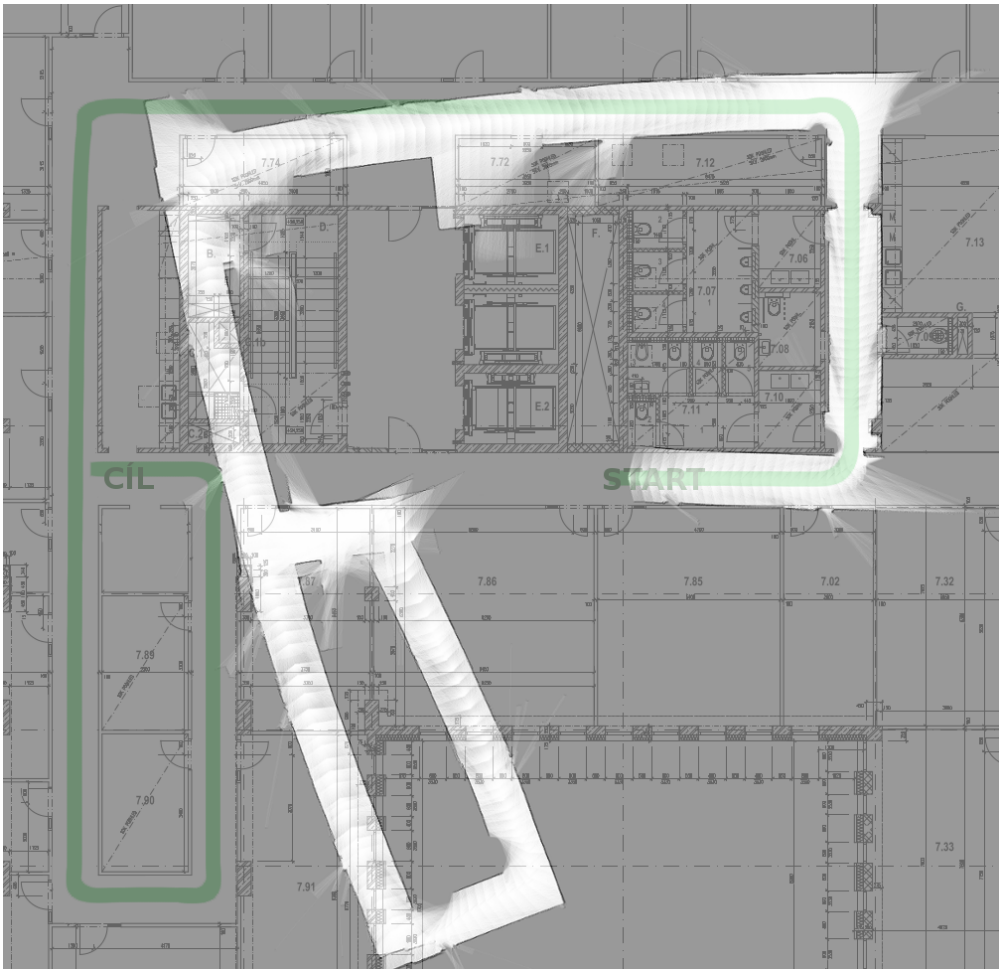
(a) (b)

Obrázek 6.8.: Mapování dlouhé chodby - získané mapy

V porovnání s referencí na obrázku 6.9 je vidět, že ačkoliv vytvářená mapa zpočátku velmi přesně mapa kopíruje skutečnost co se úhlů i vzdáleností týče, s ujetou vzdáleností po dlouhé chodbě narůstá zakřivení chodby. Na druhou stranu, i přes zakřivení chodby ke konci mapování bylo

pro toto nastavení správně detekováno uzavření smyčky. Jedním z důvodů zakřivení mapy rovné chodby je patrně opět omezený dosah použitého laserového dálkoměru. Zpočátku i zanedbatelná chyba ve vložení scanu do mapy způsobí po několika desítkách metrů odchylku od skutečnosti. Drobný vliv má také diskretizace při používání mřížky obsazenosti. Pokud se v mřížce obsazenosti hrana překážky nalézá např. na kraji buňky, je měněna pravděpodobnost celé buňky. Pro ověření vlivu jsme provedli další měření při dvojnásobném zvýšení rozlišení (0.005m). Změna v získané mapě však byla minimální a lze tedy konstatovat, že vliv diskretizace při použitém rozlišení (0.02m) můžeme zanedbat.

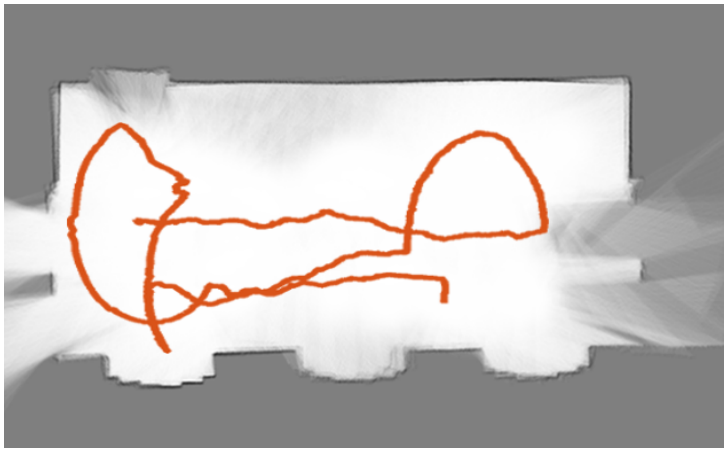
Pro porovnání byl spuštěn nad naměřenými daty také původní program používající pouze laserový dálkoměr s totožným nastavením (viz obrázek 6.7b). Dle předpokladů na dlouhých chodbách algoritmus opravdu selhal, jak je vidět z opravdu výrazného zkrácení všech chodeb.



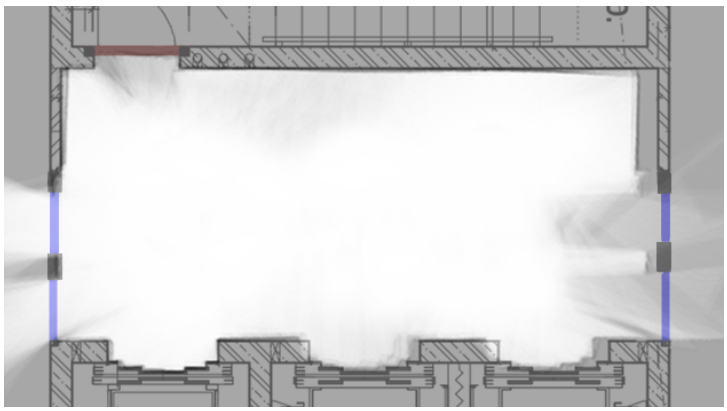
Obrázek 6.9.: Získaná mapa v porovnání s referenční technickou dokumentací.

6.3.2. Vstup k výtahům

Vstupní místost k výtahům je prostor s výrazně menší rozlohou a vyšší členitostí než v předchozím experimentu 6.3.1. Podlahu zde tvoří broušené teraso, které svým nepravidelným vzorem nabízí lepší podmínky pro videoodometrické měření (viz obrázek 6.4b). Pro mapování bylo použito nejlepších nalezených parametrů algoritmu ICP-SLAM z experimentu 6.3.1, s tím, že bylo kvůli menšímu prostředí použito dvojnásobné rozlišení mřížky obsazenosti (0.01m). Takto získaná mapa je zobrazena na obrázku 6.10.



Obrázek 6.10.: Získaná mapa vstupního prostoru k výtahům. Červenou barvou byla vykreslena zaznamenaná trajektorie kvadrokoptéry. Mapování začínalo na levé straně.



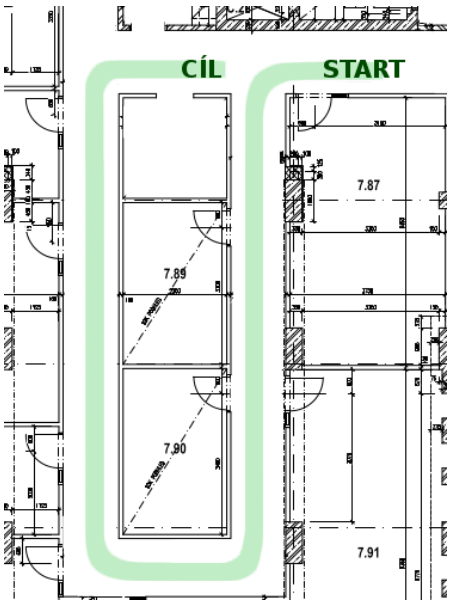
Obrázek 6.11.: Porovnání získané mapy s technickou dokumentací, modrou barvou jsou zdůrazněny skleněné průhledy na chodbu.

Z porovnání s referencí na obrázku 6.11 je vidět, že vstupní místnost k výtahům byla zmapována relativně správně, přesto je však patrná určitá chyba v délce chodby. Na základě technické dokumentace můžeme vyčíst, že chyba v celkové délce vytvořené mapy je zhruba 20cm. Malé zkrácení je patrné také u výtahových šacht na pravé straně. Pokud vezmeme v úvahu udávanou přesnost laserového dálkoměru (pro vzdálenosti delší než 1m výrobce udává přesnost ±5.6 cm) a

odhad chyby odometrie v tomto prostředí, lze považovat tuto mapu za relativně dobrý výsledek. Pokud navíc vezmeme v potaz trajektorii kvadrokoptéry, je vidět, že velmi přesně je zmapována levá část místnosti, kde se kvadrokoptéra pohybovala delší dobu a začína svůj pohyb. Zakřivování mapy, které se projevovalo při předchozím experimentu v dlouhých chodbách, není v tomto případě znatelné.

6.4. Experiment s kvadrokoptérou

Kvadrokoptéra, představená v podkapitole 5.1, nebyla bohužel v době závěrečných experimentů schopna letu stabilního natolik, aby bylo možné provést delší let ve vnitřním prostředí a získat potřebná data. Proto byl let kvadrokoptéry pouze simulován, podobně jako v předchozích případech. Navíc však byla simulována také proměnlivá výška kvadrokoptéry a její náklony (viz obrázky 6.13, 6.14). Mapování probíhalo opět v chodbách, naznačená trasa mapování je znázorněna na obrázku 6.12.



Obrázek 6.12.: Trasa mapování chodby kvadrokoptérou.

Při experimentu bylo použito nejlepší nalezené nastavení ICP-SLAM algoritmu z části 6.3.1 stejně jako mřížka obsazenosti s totožným rozlišením (0.02m). Provedená měření pro různá nastavení kompenzace náklonů jsou shrnuta v tabulce 6.4.1.

Parametry `roll_min`, `roll_max` (`pitch_min`, `pitch_max`) určují mezní hodnoty úhlu roll (pitch), pro které se ještě provede kompenzace náklonu, pro větší výchylky je laserový scan ignorován. Parametr `roll_zero_tolerance` (`pitch_zero_tolerance`) definuje ± počet stupňů od nuly, pro které se kompenzace náklonu také neprovede, ale zároveň je laserový scan v ICP-SLAM algoritmu použit bezezměny.

Obrázek 6.13.: Výška kvadrokoptéry nad podlahou v průběhu mapování.

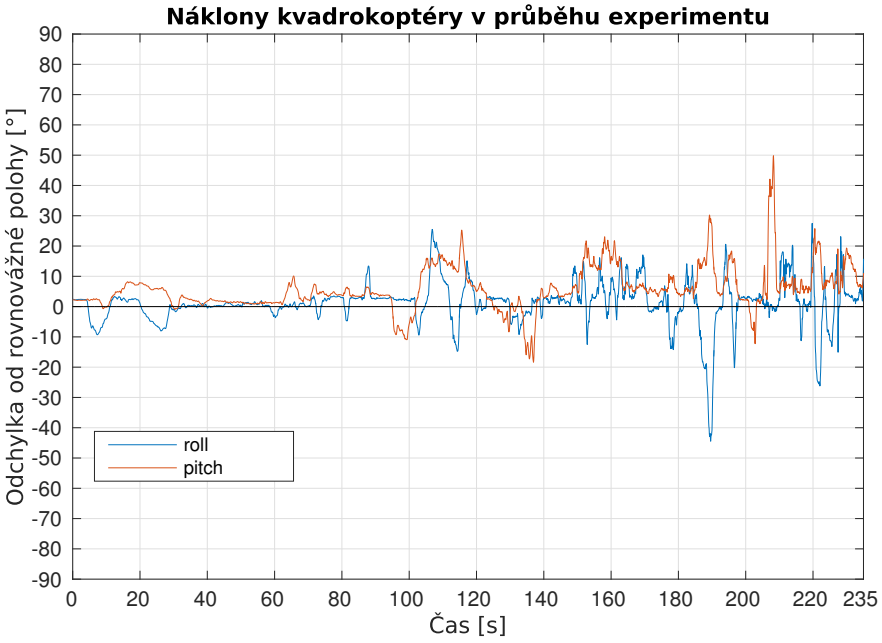
	Číslo měření:	**1**	**2**	**3**	**4**
	Obrázek s výslednou mapou	6.15a	6.15b	6.15c	6.15d
Hodnoty parametrů	Použití kompenzace náklonů	NE	ANO	ANO	ANO
	`roll_min` [°]	-	-30	-40	-40
	`roll_max`[°]	-	30	40	40
	`pitch_min`[°]	-	-210	-220	-200
	`pitch_max`[°]	-	-160	-140	-140
	`roll_zero_tolerance`[°]	-	4	6	4
	`pitch_zero_tolerance`[°]	-	4	6	4

Tabulka 6.4.1.: Experiment s kvadrokoptérou - nastavení kompenzace náklonů.

Na obrázku 6.15a je uvedena pro srovnání získaná mapa bez použití kompenzace náklonů. Mapa překvapivě dobře odpovídá šířkou chodby realitě, mapování však úplně selhává při prudkých změnách orientace kvadrokoptéry ke konci mapování. Ukazuje se tedy, že i samotná mřížka obsazenosti zvládá relativně dobře pohlcovat šum způsobený menšími náklony, pokud trvají kratší dobu (jako po většinu mapované trasy v tomto experimentu).

Příliš nízké hodnoty parametrů byly zvoleny v případě mapy na obrázku 6.15b. Mapování selhává opět ke konci při prudkých změnách orientace. Kompenzace náklonů zde smazala část měření, která byla pořízena pro výchylky úhlů roll a pitch nad definované meze a kvůli tomu došlo k částečné ztrátě informace a ke konci mapování opět selhává.

Lepší výsledek je uveden na mapě na obrázku 6.15c, zde byly mezní hodnoty úhlů zvýšeny o 10° a kromě několika měření, takřka všechny laserové scany byly zahrnuty do ICP-SLAM algoritmu,



Obrázek 6.14.: Náklony kvadrokoptéry v průběhu mapování.

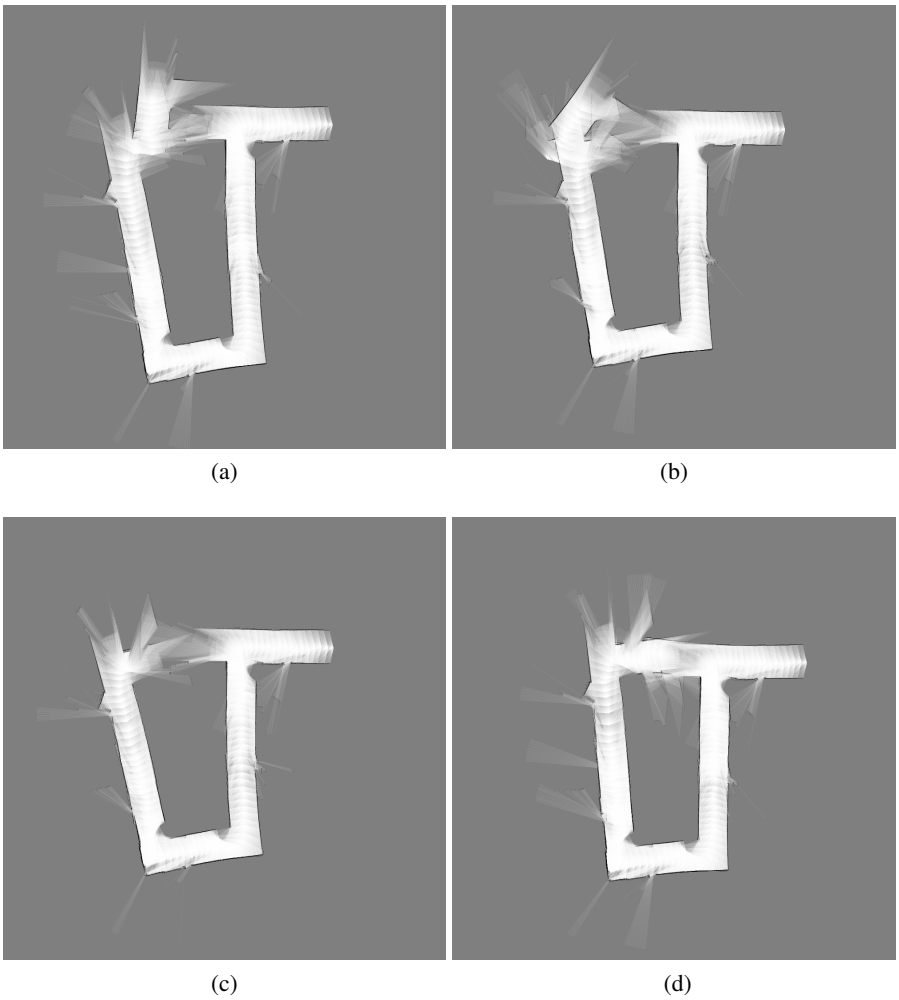
jako příliš velká hodnota se však ukázala tolerance ±6° pro vynechání kompenzace náklonů.

Pro získání mapy na obrázku 6.15d byly tyto tolerance sníženy. Vlivem toho byla kompenzace náklonů provedena pro podstatně více laserových scanů. Zde je znát výrazné zlepší ke konci mapované trasy, kde mapování bez kompenzace náklonů selhává. Srování této mapy s referencí je uvedeneno na obrázku 6.16.

U všech získaných map v experimentu s kvadrokoptérou lze vypozorovat výrazné zkrácení rovných úseků chodeb ve srovnání s předchozím experimentem ve stejném prostředí (viz část 6.3.1). Videoodometrie sice po celou dobou experimentu fungovala, nicméně integrovaná vzdálenost byla kratší. Možným důvodem zkrácení, kromě celkově horší ICP korekce vlivem šumu způsobeného náklony a otřesy, může být také to, že vzhledem ke snaze simulovat různé náklony kvadrokoptéry byl pohyb celkově podstatně méně plynulý než při experimentu s testovací platformou. Jak bylo již zmíněno v části 5.4.5, měření modulu Px4flow také částečně závisí na rychlosti pohybu. Svůj vliv má patrně také odlišná intenzita osvětlení při tomto experimentu (kvadrokoptéra byla držena v ruce a tedy částečně zastíněná).

6.5. Celkové zhodnocení experimentů

Provedené experimenty ukázaly výrazné zlepšení ve srovnání s původní ukázkovou implementací `icp-slam-live`. Navržené řešení vytváří 2D mapy vnitřního prostředí. Implementovaná videoodometrie významně snižuje zkracování chodeb způsobené selháváním ICP algoritmu pro



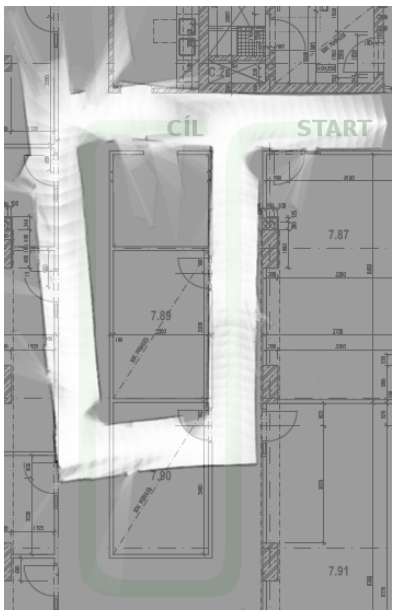
(a) (b)

(c) (d)

Obrázek 6.15.: Experiment s kvadrokoptérou - získané mapy

použitý laserový dálkoměr. Pokud bychom uvažovali mapování členitějšího prostředí, poskytuje naše řešení relativně dobré výsledky s ohledem na přesnost použitých senzorů. Při mapování chodeb se podařilo správně rozpoznat i návrat na stejné místo (uzavření smyčky), což je obecně pro scan-matching algoritmus bez dalších vylepšení obtížný úkol [24]. Vzhledem k použitému k dalším provedeným experimentům to však není vždy pravidlem, ale spíše šlo o souhrou vhodného nastavení a dobrých podmínek v testovaném prostředí. Navržená kompenzace náklonů dokázala zlepšit výslednou mapu, ačkoliv i samotný algoritmus ICP-SLAM využívající mřížku obsazenosti relativně dobře pohlcuje krátkodobé a menší náklony kvadokoptéry. Navržené řešení však trpí také několika nedostatky.

Implementovaná videoodometrie je závislá na podmínkách v mapovaném prostředí. Změny osvětlení, povrch podlahy i rychlost a plynulost pohybu mají dopad na výslednou naměřenou vzdálenost. Vlivem toho dochází k určitému zkracování rovných úseků.



Obrázek 6.16.: Experiment s kvadrokoptérou - porovnání získané mapy s referencí.

Další vlastností navrženého řešení je zakřivování rovných chodeb, které přetrvává pro všechna testovaná nastavení ICP-SLAM algoritmu. Předpokládáme, že tento problém by se částečně vyřešil použitím lepšího laserového dálkoměru, který by disponoval delším dosahem. Na většině mapovaných míst by tak patrně ani nenastávala nutnost používat videoodometrii.

Kapitola 7.

Závěr

V této práci jsme navrhli a realizovali řešení pro mapování vnitřního prostředí autonomní helikoptérou. V souladu se zadáním práce byly realizovány následující úkoly:

- Seznámili jsme s problematikou mapování vnitřního prostředí v mobilní robotice.
- Definovali jsme omezující podmínky pro řešitelnost práce.
- Vybrali jsme vhodné senzorické systémy, které jsme instalovali na helikoptéru.
- Navrhli a implementovali jsme metodu mapování vnitřního prostředí, založenou na ICP-SLAM algoritmu.
- Celé navržené řešení bylo experimentálně otestováno v několika typech vnitřního prostředí.

V teoretickém úvodu práce byl uveden ucelený rozbor problematiky včetně používaných metod mapování, senzorických systémů či stavu problematiky ve světě.

Nakonec bylo zvoleno řešení založené na scan-matching ICP-SLAM algoritmu s využitím svobodných knihoven MRPT. Jako primární zdroj senzorických dat jsme vybrali laserový dálkoměr, doplněný o modul Px4flow (kamera měřící optický tok) a inerciální měřící jednotku. Kromě senzorů byl na upravenou helikoptéru (kvadrokoptéru) z modelářských dílů instalován také kompaktní počítač Nvidia Jetson TK1, který umožňuje běh navržené metody v reálném čase přímo na kvadrokoptéře.

Pro odhad pozice kvadrokoptéry v prostředí, kde lokalizace založená pouze na laserovém dálkoměru selhává, byla dále implementována videoodometrie využívající data z modulu Px4flow a inerciální měřící jednotky. Odhady náklonů z interciální měřící jednotky byly také použity pro jednoduchou kompenzaci chyb měření laserového dálkoměru, které vznikají při náklonech kvadrokoptéry při letu.

Představené řešení vytváří 2D mapy vnitřního prostředí, za předpokladu splnění určitých podmínek. Mezi ně patří kromě správného nastavení ICP-SLAM algoritmu především vysoká intenzita osvětlení, vhodný povrch podlahy a plynulost celého pohybu kvadrokoptéry.

7.1. Možnosti dalšího rozšíření práce

Celá tato práce byla vytvářena s předpokladem budoucí plně autonomní činnosti kvadrokoptéry ve vnitřním prostředí. Pro dosažení plné autonomnosti při mapování je však nutné stále vyřešit několik následujících bodů.

Co se týče navrženého softwarového řešení mapování, další vývoj by měl směřovat především ke zvýšení kvality sestavované mapy. Jednou z možností je implmentovat průběžnou optimalizaci mapy založenou na grafu poloh pro detekci uzavírání smyček, podobně jako v práci [24]. Pro vytváření velkých a přesnějších map bude zřejmně nutné alespoň částečně eliminovat fenomén „zakřivování“ dlouhých chodeb. Za úvahu zde stojí především vyzkoušet lepší laserový dálkoměr s delším dosahem. Dalším možností je pokusit se zmenšit závislost dat z modulu Px4flow na podmínkách v mapovaném prostředí (např. instalací vhodného osvětlení na kvadrokoptéru). Za úvahu jistě stojí také využití velké mobility kvadrokoptéry pro rozšíření mapování do 3D.

Po hardwarové stránce bude pro praktické použití nezbytné instalovat na kvadrokoptéru WiFi modul pro přenos informací k uživateli v reálném čase a zprovoznit obousměrnou komunikaci počítače Jetson TK1 a Desky regulátorů rychlostí, tak aby bylo možné z počítače řídit pohyb kvadrokoptéry. Potřebné sériové rozhraní již bylo připraveno v rámci této práce.

Aby byl možný bezpečný provoz i ve složitějším vnitřním prostředí, je nutné výrazně zlepšit celkové letové vlastnosti kvadrokoptéry. Za zvážení stojí případná instalace dalších senzorů pro robusní detekci překážek a montáž mechanické ochrany vrtulí.

Za předpokladu splnění těchto bodů může být kvadrokoptéra v budoucnu použita pro implementaci dalších algoritmů, např. pro autonomní exploraci prostředí, plánování tras ap.

Literatura

[1] *Velodyne Lidar - Puck Lite* [online]. 2016. Dostupné z: <http://velodynelidar.com/vlp-16-lite.html>.

[2] *Hokuyo URG-04LX Product specification* [online]. 2005 - 2016. Dostupné z: http://www.hokuyo-aut.jp/02sensor/07scanner/URG-04LX_spec_en.pdf.

[3] *MAVLink Micro Air Vehicle Communication Protocol* [online]. 2016. Dostupné z: <http://qgroundcontrol.org/mavlink/start>.

[4] *Mobile Robotics Programming Toolkit* [online]. (c) 1996-2016. Dostupné z: <http://www.mrpt.org>.

[5] Amato, A. *5 Movies Filmed With Drones* [online]. 25.9.2014. Dostupné z: <http://dronelife.com/2014/09/25/movies-scenes-shot-with-drones/>.

[6] Augugliaro, F. et al. Building tensile structures with flying machines. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, s. 3487. IEEE, 2013.

[7] Bednařík, M. *Fyzika 1*. České vysoké učení technické, 2012.

[8] Bedroš, R. *Modul lokalizace mobilního robotu pro systém Player*. Bakalářská práce, České vysoké učení technické, Praha, 2011.

[9] Blanco, J. L. *MRPT application: icp-slam-live* [online]. 18.7.2015. Dostupné z: <http://www.mrpt.org/list-of-mrpt-apps/application-icp-slam-live/>.

[10] Burian, M. *Environment mapping using radar*. Bakalářská práce, České vysoké učení technické, Praha, 2014.

[11] Báča, T. *Model predictive control of micro aerial vehicle using onboard microcontroller*. Diplomová práce, České vysoké učení technické, Praha, 2015.

[12] Choudhary, S. Simultaneous Localization and Mapping: Literature Survey. 2014.

[13] Christensen I. Henrik, H. G. D. Sensing and estimation. In Bruno Siciliano, O. K. (Ed.) *Springer handbook of robotics*. Springer Science & Business Media, 2008. 4, s. 88–107.

[14] D. Michael Sobers, E. N. J. G. C. Indoor Navigation for Unmanned Aerial Vehicles. In *AIAA Guidance, Navigation, and Control Conference, Chicago, Illinois*. American Institute of Aeronautics and Astronautics, 2009.

[15] Daniel Sack, W. B. A comparison of methods for line extraction from range data. 2004.

[16] Fossel, J. et al. OctoSLAM: A 3D mapping approach to situational awareness of unmanned aerial vehicles. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, s. 179–188. IEEE, 2013.

[17] Grzonka, S. – Grisetti, G. – Burgard, W. A fully autonomous indoor quadrotor. *Robotics, IEEE Transactions on*. 2012, 28, 1, s. 90–100.

[18] Henry, P. et al. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*. 2012, 31, 5, s. 647–663.

[19] Honegger, D. et al. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, s. 1736–1741. IEEE, 2013.

[20] Huh, S. – Shim, D. H. – Kim, J. Integrated navigation system using camera and gimbaled laser scanner for indoor and outdoor autonomous flight of UAVs. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, s. 3158–3163. IEEE, 2013.

[21] Kim, J.-H. – Sukkarieh, S. Airborne simultaneous localisation and map building. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, 1, s. 406–411. IEEE, 2003.

[22] Kubík, P. *Semiadaptivní 3D modeling*. Diplomová práce, Univerzita Karlova v Praze, Praha, 2006.

[23] Kulich, M. *Lokalizace a tvorba modelu prostředí v inteligentní robotice*. Disertační práce, České vysoké učení technické, Praha, 2003.

[24] Kumar Vijay, N. S. Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV. In *IEEE International Conference on Robotics and Automation (IROS)*, s. 20–25. IEEE, IEEE, 2011.

[25] L. Hájek, T. H. Gyroskopy. 2010.

[26] Langelaan, J. – Rock, S. Passive GPS-free navigation for small UAVs. In *Aerospace Conference, 2005 IEEE*, s. 1–9. IEEE, 2005.

[27] Martin Adams, E. J. B.-N. V. J. M. *Robotic Navigation and Mapping with Radar*. Artech House Inc, 2012. ISBN 978-1-60807-482-2.

[28] Matteo Golfarelli, S. D. Elastic Correction of Dead-Reckoning Errors in Map Building. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on Intelligent Robots and Systems*, s. 905–911. IEEE, 1998.

[29] Montemerlo, M. et al. FastSLAM: A factored solution to the simultaneous localization and mapping problem. 2002.

[30] Mynařík, O. *Využití mobilních zařízení ve zpětnovazební lékařské rehabilitaci*. Bakalářská práce, Vysoké učení technické v Brně, Brno, 2015.

[31] Mázl, R. *Lokalizace pro autonomní systémy*. Disertační práce, České vysoké učení technické, Praha, 2007.

[32] Omead Amidi, R. T. Vision-based Autonomous Helicopter Research at Carnegie Mellon Robotics Institute 1991-1997. In *IEEE International Conference on Robotics and Automation (IROS)*.

[33] Riisgaard, S. – Blas, M. R. SLAM for Dummies. *A Tutorial Approach to Simultaneous Localization and Mapping*. 2003, , 1-127, s. 17.

[34] Scherer Sebastian A., K. Efficient Onbard RGBD-SLAM for Autonomous MAVs. In *IEEE International Conference on Robotics and Automation (IROS)*, s. 1062–1068. IEEE, IEEE, 2013.

[35] Stanislav Daďo, M. K. *Senzory a měřící obvody*. České vysoké učení technické, 1996.

[36] Thrun, S. Robotic Mapping: A Survey. 2002.

[37] Thrun Sebastian, L. J. J. Simultaneous Localization and Mapping. In Bruno Siciliano, O. K. (Ed.) *Springer handbook of robotics*. Springer Science & Business Media, 2008. 37, s. 871–886.

[38] Wang Fei, C. L. C. A Comprehensive UAV Indoor Navigation System Based on Vision Optical Flow and Laser FastSLAM. In *50th Anniversary of Acta Automatica Sinica*, s. 1890–1898. Chinese Association of Automation, Chinese Association of Automation, 2013.

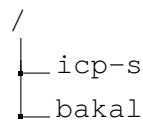
[39] Wolfram Burgard, M. H. World modeling. In Bruno Siciliano, O. K. (Ed.) *Springer handbook of robotics*. Springer Science & Business Media, 2008. 36, s. 853–867.

[40] Šiška, M. *Impulzové modulace*. Diplomová práce, Vysoké učení technické v Brně, Brno, 2013.

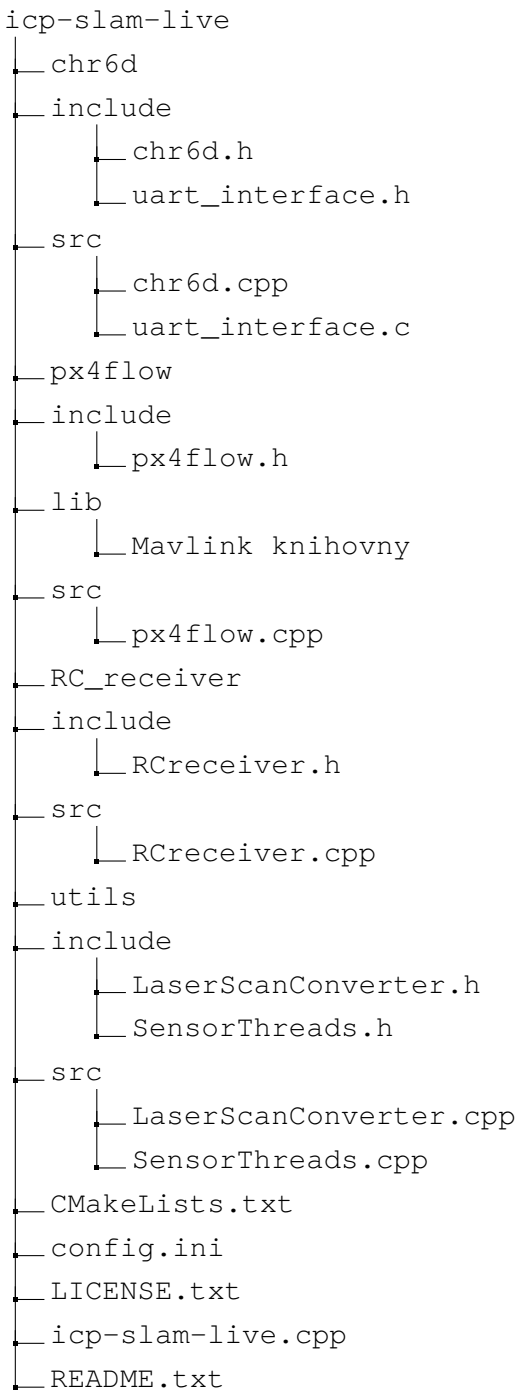
Přiloha A.

A Obsah CD

Obsah kořenového adresáře na CD je uveden na obrázku A.1, obsah adresáře `icp-slam-live` s implementací mapování je na následující straně na obrázku A.2.



Obrázek A.1.: Obsah přikládaného CD



Obrázek A.2.: Obsah adresáře `icp-slam-live` s implementací mapování

Příloha B.

Konfigurační .ini soubor

[LASER]

driver = CHokuyoURG //jmeno pouziteho driveru z knihovny MRPT

process_rate = 10 // maximalni snimaci frekvence dane laseru (lze snizit)

COM_port_LIN = ttyACM1 //misto pripojeni laseru v OS Linux

sensorLabel = LASER

#Pozice laseroveho dalkomeru vzhledem k telu robotu:

pose_x = 0.05

pose_y = 0

pose_z = 0

pose_yaw = 180 //vse ve stupnich:

pose_pitch = -180

pose_roll = 0

reduced_fov = 220 //omezeni maximalniho uhloveho rozsahu laseru

#============================

[PX4FLOW]

COM_port_LIN = ttyACM0 //misto pripojeni modulu Px4flow v OS Linux

sensorLabel = px4flow

baudrate = 115200

verbose = false //ladici vypisy vlakna parsujiciho Mavlink pakety

#============================

[CHR6D]

COM_port_LIN = ttyUSB1 //misto pripojeni IMU v OS Linux

sensorLabel = chr6d

baudrate = 115200

#Pouzivat ve videoodometrii integraci uhlove rychlosti z gyroskopu?

estimateYawDisplacement = false

#============================

[RCreceiver]

#Pouzivat UART komunici mezi PC aRC prijimacem pres Deskou regulatoru?

#->Zatim je zprovozneno spousteni a zastaveni programu pres RC vysilacku.

useRCreceiver = false

COM_port_LIN = DOPLNIT

baudrate = = 57600

#============================

[LaserScanConverter]

#Pouzivat kompenzaci naklona kvadrokoptery?

useLaserScanConverter = true

verbose = false // vypisovani ladicich vystupu kompenzace

#Min. vzdalenost od zeme, pro zahrnuti laseroveho scanu do ICP-SLAM algoritmu.

#Slouzi pro filtrovani mereni pri startu kvadrokoptery, -1 tuto moznost vypina.

height_min = -1

#Mezni hodnoty pro zahrnuti laseroveho scanu do ICP-SLAM algoritmu a provedeni

#kompenzace naklonu, hodnoty jsou ve °.

roll_min = -40

roll_max = 40

#v rovnovazne poloze je uhel pitch -180° (vzhledem ke zvolene vztazne soustave)

pitch_min = -160 //-180 + ...

pitch_max = -200 //-180 - ...

#Tolerance pro vynechani kompenzace naklonu (tj. predpoklad rovnovazne polohy),

+- od od rovnovazne polohy.

pitch_zero_tolerance = 4

roll_zero_tolerance = 4

#============================

[VideoOdometry]

Pouzivat videoodometrii?

useOdometry = true

verbose = false

#=========================

[ICP]

Maximalni pocet iteraci, ktere se vykonaji, pokud nebylo konvergence dosazeno

#drive.

maxIterations = 80

#Pokud je oprava ve vsech posunech (X,Y,Z) pod touto hranici, ICP je zastaveno.

minAbsStep_trans = 1e-6 //v metrech

#Pokud je oprava ve vsech uhlech (yaw,pitch,roll) pod touto hranici, ICP je zastaveno.

minAbsStep_rot = 1e-6 // v radianech

#Pocatecni maximalni vzdalenost pro oznaceni dvojice bodu jako „bodovy par“.

thresholdDist = 0.20

#Pocatecni maximalni uhlova vzdalenost pro oznaceni bodu jako „bodovy par“.

thresholdAng_DEG = 5 // ve °

#Konstanta, kterou jsou vynasobeny limity po dosazeni konvergence, coz udrzuje

#beh ICP

#algoritmu a poskytuje presnejsi transformaci.

ALFA = 0.5

#Nejmensi hodnota limitu vzdalenosti, kterou muze treshold nabyvat

#po zastaveni ICP

#algoritmu a prijeti vysledku.

smallestThresholdDist=0.04

#true: bodovy par = nejblizsi korespondujici body

#false: pouziji se vsechny korespondence v ramci tresholdu (pomalejsi)

onlyClosestCorrespondences=true

#Vynutit unikatni korespondence bodu v obou smerech parovani?

onlyUniqueRobust = false

#Jaky typ ICP algoritmu pouzit?

#Klasicke icp dava lepsi a robusnejsi vysledky

icpClassic

icpLevenbergMarquardt , nepodporuje pasovani na mrizku obsazenosti

ICP_algorithm = icpClassic

#=========================

[MappingApplication]

nazev adresare s vytvarenymi soubory

logOutput_dir=LOG_SLAM

#Frekvence vytvareni souboru

LOG_FREQUENCY=50

*.3Dscene je format MRPT knihoven pro pozdejsi vyzualizaci pohybu

SAVE_3D_SCENE=1

#Ukladat prubezne pozice robotu jako kovariancni matice?

SAVE_POSE_LOG=1

#nastaveni GUI:

CAMERA_3DSCENE_FOLLOWS_ROBOT=1

SHOW_PROGRESS_3D_REAL_TIME=1

SHOW_PROGRESS_3D_REAL_TIME_DELAY_MS=5

SHOW_LASER_SCANS_3D = true

Ukladani vsech porizenych laserovych scanu + videoodometrie ?

SAVE_RAWLOG = false

Ukladani pouzitych laserovych scanu + videoodometrie ?

SAVE_RAWLOG_SPARSE = true

#Min. hodnoty pro provedeni ICP korekce pozice:

localizationLinDistance = 0.02 // v metrech

localizationAngDistance = 2 // ve °

#minimalni hodnoty nezbytne pro vlozeni do mapy:

insertionLinDistance = 0.25 // v metrech

insertionAngDistance = 8 //ve °

#Minimalni kvalita ICP algoritmu pro provedeni korekce (vlozeni do mapy),

#% hodnota jako cislo mezi <0, 1>

minICPgoodnessToAccept = 0.40

#Pouzivat mrizku obsazenosti?

#Nezbytne pro LM metodu, ktera jako jedina podporuje pasovani primo

#na bodovou mapu

matchAgainstTheGrid = 1

Jake mapy se maji krome SimpleMap vytvorit po ukonceni programu?

occupancyGrid_count=1

pointsMap_count=1

#=========================

[MappingApplication_pointsMap_00_insertOpts]

minDistBetweenLaserPoints = 0.02

fuseWithExisting = false

Jde o 2D mapovani?

isPlanarMap = 1

#============================

[MappingApplication_occupancyGrid_00_creationOpts]

#Rozliseni mrizky obsazenosti

resolution=0.02

disableSaveAs3DObject=0

#============================

[MappingApplication_occupancyGrid_00_insertOpts]

nastaveni pro 3D mapovni:

mapAltitude=0

useMapAltitude=false

//nastaveni pro pro pouzity laserovy dalkomer:

maxDistanceInsertion=8

maxOccupancyUpdateCertainty=0.54

considerInvalidRangesAsFreeSpace=1

minLaserScanNoiseStd=0.001