ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta strojní – Ústav přístrojové a řídicí techniky

Navigace zemědělského robota v sadu

Diplomová práce

Grebennikov Pavel 2024

Poděkování

V prvé řadě bych rád poděkoval svému vedoucímu diplomové práce Ing. Mgr. Jakubu Jurovi, Ph.D. za rady, odbornou pomoc a trpělivost. Dále bych rád poděkoval všem účastníkům projektu, kterého jsem se mohl účastnit. Jmenovitě bych vyzdvihnul pomoct pana Ing. Jakuba Lva, Ph.D. za konzultaci a rady s využitím knihovny *OSGAR*, a pana Ing. Matouše Cejnka, Ph.D. za poskytnutí natrénovaného modelu pro detekci kmenů a rady s jeho praktickým využitím. V neposlední řadě moc děkuji Mgr. Pavle Hanzalové, Ph.D. za pomoc s korekturou formální stránky práce a jazykovou kontrolu.

Prohlašuji, že jsem tuto práci vypracoval samostatně s použitím literárních zdrojů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů.

Datum: . . . .

Podpis

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Grebennikov** Jméno: **Pavel** Osobní číslo: **484715**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídící techniky**

Studijní program: **Automatizační a přístrojová technika**

Specializace: **Automatizace a průmyslová informatika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Navigace zemědělského robota v ovocném sadu

Název diplomové práce anglicky:

Agricultural robot navigation in the orchard

Pokyny pro vypracování:

Hlavním cílem práce je vytvoření systému navigace robota v ovocném sadu (koncepce Agriculture 4.0). Dílčí úkoly jsou:
1) Rešerše robotických map
2) Fúze senzorů robota (lidar, odometrie, GPS, kamery) v robotické mapě
3) Korigování obsluhy robota při manuálním řízení včetně HMI v režimu poloautonomního snímkování
4) Automatické vedení robota v řádce, automatická otočka a zastavování
5) Automatická identifikace jednotlivých stromů a registrace údajů v robotické mapě

Seznam doporučené literatury:

L. Payä, F. Amorós, L. Fernández, O. Reinoso. An educational software to develop robot mapping and localization practices using visual information, IFAC Proceedings Volumes, Volume 46, Issue 17, 2013, ISSN 1474-6670, ISBN 9783902823434, https://doi.org/10.3182/20130828-3-UK-2039.00043.
(https://www.sciencedirect.com/science/article/pii/S1474667015340969)
OSGAR GIT, https://github.com/robotika/osgar

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Mgr. Jakub Jura, Ph.D. U12110.3

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **27.10.2023** Termín odevzdání diplomové práce: **19.01.2024**

Platnost zadání diplomové práce: _____

Ing. Mgr. Jakub Jura, Ph.D.	prof. Ing. Tomáš Vyhlídal, Ph.D.	doc. Ing. Miroslav Španiel, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

Abstrakt

Tato práce se zaměřuje na implementaci konceptu Zemědělství 4.0 v ovocném sadu. Klíčovým prvkem je zde fúze senzorů zahradního robota pro tvorbu robotické mapy. Teoretická část se zaměřuje na popis konceptu Zemědělství 4.0 z hlediska využitých technologií, využitých přístupů a praktických aplikací. Druhá část popisuje obecné přístupy k fúzi senzorických dat a obsahuje rozdělení jednotlivých metod doplněných o jejich stručný popis. Poslední část pojednává o robotickém mapování, jeho výzvách, algoritmech a typech map.

Praktická část se zabývá zpracováním roboticky naměřených senzorických dat. Data z kamer, senzorů GPS, enkodérů a lidaru jsou zde využita pro detekci jednotlivých stromů. Součástí práce je i vyvinutý program v prostředí Python, jehož výstupem je robotická mapa prostředí. Ke každému detekovaném stromu v mapě jsou přiděleny jeho fotografie a informace o jeho relativní a absolutní poloze. Další část se zabývá režimem poloautonomního snímkování doplněného o vývoj uživatelského rozhraní zobrazujícího informace klíčové pro operátora vozidla. Poslední část obsahuje konceptuální návrh autonomních prvků řízení robota včetně vedení v řadě, zastavování a otočky.

Klíčová slova: Zemědělství 4.0, fúze senzorů, robotické mapování, detekce stromů.

Abstract

This thesis focuses on the implementation of Agriculture 4.0 concepts in an orchard setting with key emphasis on the sensor fusion of a garden robot for the creation of robotic map. The theoretical section provides an in-depth exploration of the Agriculture 4.0 concept, examining the technologies employed, approaches adapted and practical applications. The second part outlines general strategies for sensor data fusion, presenting a classification of various methods accompanied by concise descriptions. The final section addresses the intricacies of robotic mapping, discussing its challenges, algorithms, and several map types. The practical section revolves around processing sensor data acquired by the robot. Data from cameras, GPS sensors, encoders, and LiDaR are used for the detection of individual trees. This part of the work also includes development of a Python program designed to generate robotic map of the environment. Each identified tree on the map is associated with its photographs and information about its relative and absolute position. Next segment focuses on a semi-autonomous mode of capturing pictures complemented by the creation of a user interface displaying crucial information for the robot operator. The concluding section encompasses the conceptual design of autonomous robot control elements covering aspects such as row following, stopping, and turning.

Keywords: Agriculture 4.0, sensor fusion, robotic mapping, tree detection.

Obsah

Seznam zkratek

AE	Autoencoders
ANN	Artificial Neural Networks
CAN	Controller Area Network
CNN	Convolutional Neural Network
DBN	Deep Belief Network
DKF	Distributed Kalman Filter
DSS	Decision Support System
EIF-SLAM	Extended Information SLAM
EKF	Extended Kalman Filter
EKF-SLAM	Extended Kalman Filter SLAM
EPnP	Efficient Perspective-n-Point
FOV	Field of View
GLONASS	Globalnaya Navigatsionnaya Sputnikovaya
HLF	High-Level Fusion
HMI	Human Machine Interface
ICP	Interval Constraint Propagation
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
IoT	Internet of Things
JPDA	Joint Probability Data Association
KNN	K-Nearest Neighbours
LiDaR	Light Detection and Ranging
LLF	Low-Level Fusion
LoRaWAN	Long Range Wide Area Network
MARS	Mobile Agricultural Robot Swarm
MLF	Mid-Level Fusion
NFC	Near Field Communication
OSGAR	Open Source Garden Autonomous Robot
PaaS	Platform as a Service
PF-SLAM	Particle Filter SLAM
PnP	Perspective-n-Point
RANSAC	Random Sample Consensus
RBPF	Rao-Blackwellized Particle Filter
RFID	Radio-Frequency Identification
RGB	Red Green Blue
RHEA	Robot fleets for Highly Effective Agriculture

RNN	Recurrent Neural Network
RTK	Real Time Kinematic
RTK-GNSS	Real Time Kinematic - Global Navigation Satelite System
SaaS	Software as a Service
SIR	Sampling Importance Resampling
SLAM	Simultaneous Localization and Mapping
SOA	Service Oriented Architecture
SPLAM	Simultaneous Planning, Localization and Mapping
UAV	Unmanned Aerial Vehicles
UGV	Unmanned Ground Vehicles
UI	User Interface
VPN	Virtual Private Network
Wi-Fi	Wireless Fidelity
WSN	Wireless Sensor Networks
YOLO	You Look Only Once

Úvod

Je všeobecně známo, že zemědělství je životně důležitou činností pro lidské přežití. Kromě potravy nám slouží k získávání biopaliv, rostlinných vláken nebo surovin důležitých pro průmysl. První éra zemědělství nazývaná jako Agriculture 1.0 využívala pouze lidské nebo zvířecí síly a jednoduché ruční nástroje. S vynálezem a zefektivněním parního stroje došlo v devatenáctém století k náhradě zvířecí síly za strojní. Kromě podpory pěstovaných rostlin v podobě hnojení se začaly využívat i pesticidy zajišťující lepší sklizeň. Často byly využívány příliš extenzivně a docházelo tak ke kontaminaci prostředí a narušení ekosystému. Ve dvacátém století začala s rozvojem výpočetní techniky a elektroniky třetí etapa zemědělství. Využití počítačových programů, robotiky a mimo jiné GPS signálu, zajistilo efektivnější setí, zavlažování i sklizeň. Díky tomuto pokroku a zvýšenému ohledu vůči životnímu prostředí bylo značně zredukováno využití chemikálií. Rapidní růst lidské populace zapříčiněný zlepšením životních podmínek a lékařské péče s sebou nese požadavek na produkci stále většího a většího množství potravin. Dá se očekávat, že v budoucnosti dojde k nárůstu jevů nepříznivých pro současný způsob zemědělství, jako jsou sucha, degradace půdy a znečištění prostředí.

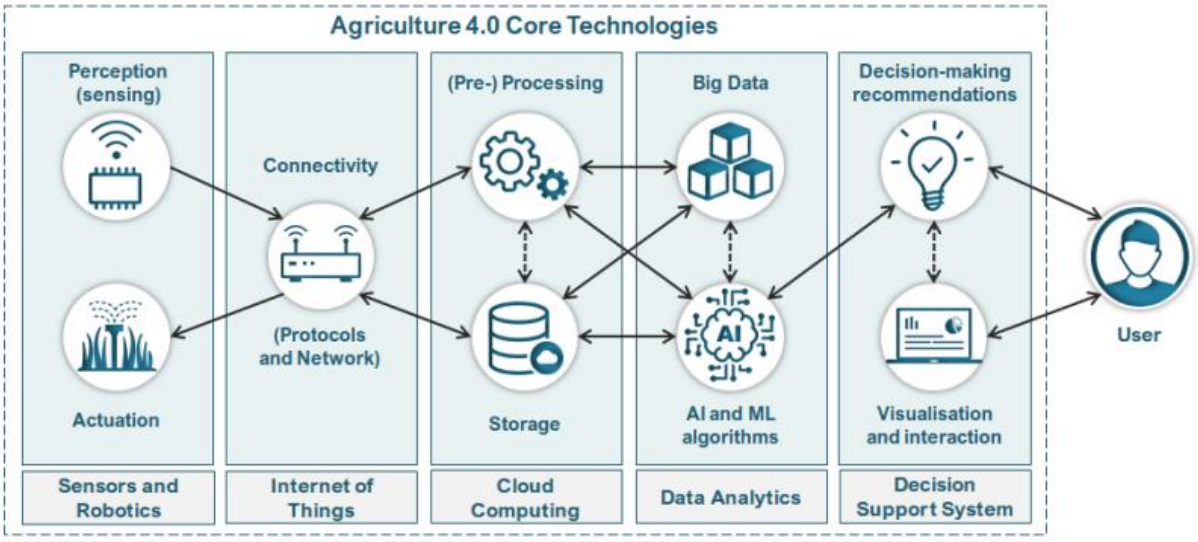
Čtvrtá etapa zemědělství, Agriculture 4.0, se začala rozvíjet přibližně v roce 2011. V tomto období byl představen pojem Industry 4.0, jehož cílem bylo vybudovat flexibilní model s využitím nejnovějších digitálních technologií pro zefektivnění produkčního řetězce. Tyto technologie jako Internet of Things (IoT), cloud computing, big data a umělá inteligence nám otevřely nové možnosti zpracování obrovských množství dat a autonomních řešení problémů. Pokročilým zpracováním dat a kombinací informací ze senzorů zemědělských robotů s technologiemi umělé inteligence a strojového učení jsme schopni zásadně zefektivnit a zvýšit produkci, zlepšit kvalitu potravin a optimalizovat celý proces pro co nejlepší udržitelnost a nejmenší dopad na životní prostředí. Cílem této práce je představit přístupy fúze senzorických informací v zemědělském prostředí se zaměřením na tvorbu robotické mapy. Následně jsou některé z představených přístupů aplikovány na konkrétní robotická měření v prostředí jablečného sadu. [1, s. 1–2], [2, s. 1]

1 Agriculture 4.0

Zemědělská produkce se odlišuje od ostatních výrobních sektorů hlavně tím, že vyžaduje významný podíl přírodního kapitálu (vzduch, půda, pozemky, …). Navíc je silně ovlivňována často nekontrolovatelnými vlivy, což dělá celé zemědělské prostředí značně proměnlivé. V důsledku těchto vlivů musí být nasazená robotická aplikace schopna dynamicky reagovat na různé struktury a charakteristiky prostředí. Samotný pojem zemědělství 4.0 představuje koncepty možné integrace moderních technologií do zemědělské produkce. [9, s. 2]

1.1 Přehled základních technologií

Využití dat v zemědělském sektoru pro optimalizaci jednotlivých procesů není nový koncept, novinkou je možnost digitalizace celého odvětví. Dalším aspektem je kvalita získaných informací a technologie využívané ke sběru, ukládání, zpracování a sdílení dat. Pokroky v senzorové oblasti umožňují sledovat specifické parametry v reálném čase. Tyto senzory jsou často umístěny přímo na robotech, což podporuje automatizaci těchto procesů. Další pokroky byly zaznamenány i v oblasti výpočetní techniky, kde výrazně stoupl výkon a zlepšila se dostupnost. To dalo za vznik technologiím, které efektivně zpracovávají velké shluky dat. Na obrázku č. 1 jsou naznačené jednotlivé druhy technologií propojené šipkami reprezentujícími tok dat.



Obrázek 1: Technologie a schéma koncepce zemědělství 4.0 [1, s. 8]

Senzorická a robotická část obsahuje fyzicky ovládaná zařízení, která vykonávají zadanou práci, a sensory poskytující požadované informace. Tyto naměřené informace jsou přes drátovou nebo bezdrátovou síť posílány dále na cloud server, kde čekají na další zpracování. Zde jsou podrobeny analýze s využitím metod big data, umělé inteligence nebo strojového učení v závislosti na aplikaci. Proces je poté zakončený rozhodovacím systémem, který z analyzovaných a přeložených dat dojde k závěru. V závislosti na míře automatizace konkrétní úlohy tyto systémy rozhodují samostatně nebo poskytnou uživateli návrhy řešení. Komunikace mezi uživatelem a podpůrným rozhodovacím systémem probíhá většinou prostřednictvím nějakého grafického rozhraní.

1.1.1 Senzory

Senzory hrají klíčovou roli při měření parametrů pro automatické řízení nebo při plnění úkolů spojených s počítačovým viděním a detekcí. Novým oborem, který využívá chytré senzory je IoT (Internet of Things). Využívá bezdrátové sítě a sensory pro sledování dat v reálném čase a

na základě vhodného zpracování a analýzy vykoná požadované akce. V posledních letech se využití senzorů značně rozšířilo do velkého množství nových odvětví. Tento růst je dán hlavně technologiemi, díky kterým jsou sensory menší, levnější a výkonnější. Snížení ceny a spotřeby energie jednotlivých senzorů dalo vzniknout tzv. chytrým senzorům. Jedná se o velký počet propojených sensorů, které monitorují danou oblast. Produkují tak obrovská množství dat, která se dají použít za účelem trénování nějaké formy umělé inteligence nebo prediktivních systémů. V oblasti zemědělství se můžeme setkat se dvěma hlavními přístupy získávání dat – vzdálené snímání nebo využití bezdrátových sítí senzorů. [4]

Vzdálené snímání (Remote Sensing) je způsob získávání informací o objektu nebo oblasti ze vzdálenosti bez přímého fyzického kontaktu. Je to nástroj pro monitorování zdrojů na zemi využívající technologie ve vesmíru s pozemními pozorováními zlepšujícími přesnost. Různé odezvy objektů vystavených elektromagnetickému záření (infračervenému a mikrovlnnému) nám slouží k rozpoznání typu vegetace, který se v dané oblasti nachází. Zároveň se tímto způsobem dají kontrolovat růst a nemoci rostlin a jiné vlastnosti jako stav vody v oblasti nebo předpověď počasí, sklizně a celkového výnosu. Pro předpověď sklizně jsou data ze vzdáleného snímkování propojena s odpovídajícími simulačními modely. Hlavními výhodami tohoto přístupu sběru informací jsou opakovatelnost měření bez destrukce produktu ve stádiu vývoje a využitelnost pro levné monitorování velkých oblastí. [6, s. 1–2]

Bezdrátové sítě senzorů WSN (Wireless Sensor Networks) jsou složeny z velkého množství senzorů, které jsou rozmístěny po snímané oblasti. Nasbíraná data jsou odesílána na centrální lokaci, kde jsou dále zpracovávána. WSN systém umí sbírat obrovská množství dat z monitorované oblasti a v případě uzavřeného systému řídí a kontroluje klimatické podmínky a kvalitu ovzduší. Kromě pozemních bateriově napájených senzorů existují i systémy operující v podzemí. V případě potřeby zisku dat z hloubky jsou senzory umístěny přímo do půdy a často napájeny přes kabely. V tomto konkrétním případě dojde ke značnému snížení vzdálenosti, na kterou jsou senzory schopny komunikovat díky nepříznivé propagaci signálu půdou, takže může dávat větší smysl využití drátových senzorů. Vzájemné propojení také značně sníží nutný počet senzorů. Příklady několika senzorů využívaných v zemědělském prostředí: [7, s. 1–4]

- meteorologické senzory,
- fotometrické senzory (měření úrovně světla),
- senzory vody,
- dendrometry (měření rozměrů rostlin – stromů),
- LIDAR (Light Detection And Ranging),
- optické kamery,
- senzory vlhkosti půdy.

1.1.2 Robotika

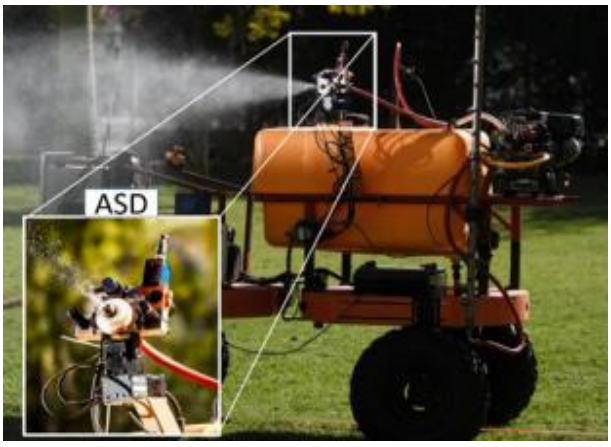
Integrování robotické technologie hlouběji do zemědělství by mohlo vytvořit nový zlom v produktivitě práce. Roboti imitací lidských dovedností nebo jejich rozšířením překonávají kritická omezení lidské práce včetně schopností v obtížných prostředích a nebezpečných podmínkách. Mají potenciál snížit dopad fyzicky náročných rutinních nebo namáhavých prací a odstranit nedostatek pracovníků pro sezónní práce. Vzhledem k proměnlivosti prostředí by roboti měli mít několik důležitých vlastností. Pokud jde o provozní nastavení, robot by měl být konfigurovatelný z hlediska polních uspořádání – velikost, tvar, typ půdy a měl by být přizpůsobitelný různým plodinám. Druhou klíčovou vlastností je bezpečný pohyb v neznámém

proměnlivém prostředí, aby zabránil poškození sebe nebo plodin. Třetí a neposlední vlastností je nastavitelná citlivost při manipulaci podle druhu produktu, aby uměl reagovat na jeho proměnlivou velikost a tvrdost. V neposlední řadě je nutná komunikace s ostatními účastníky farmářského procesu ať je to člověk nebo jiný stroj. Komunikace a spolupráce jsou nutné adresovat hlavně při velkoplošném nasazení, kde se bez nich neobejdeme. Citovaná literatura rozpoznává pět různých druhů vzájemné spolupráce: [9, s. 1–2]

Spolupráce člověk – robot

Většina zemědělských prací je v současnosti prováděna lidmi buď manuálně, nebo s pomocí strojů. Pro dosažení spolupracujícího řízení je třeba mezi člověkem a robotem vytvořit odpovídající rozhraní umožňující sdílení informací a řízení. Jeden z příkladů praktické aplikace popisuje systém s třívrstvou architekturou reprezentující servořízení, autonomní řízení a manuální provoz. Operátor tak může buď manuálně ovládat vozidlo nebo může fungovat jako dozorce automaticky ovládaného vozidla s možností vnějšího zásahu prostřednictvím rozhraní. Další příklad se zabýval automatickým rozpoznáváním melounů, kde bylo cílem zhodnotit spolupráci identifikace mezi operátorem a robotem. Testování bylo provedeno na více úrovních od čistě lidské po čistě automatickou identifikaci pro určení ideálního stavu při vzájemné spolupráci. Na téma optimalizace byla zveřejněna práce, ve které byla na vyhodnocení použita matematická programovací struktura. Pro ověření platnosti struktury byly provedeny simulace robotické sklizně citrusových plodů, ve kterých byla spolupráce s člověkem využita pro kompenzaci neefektivních složek automatizovaného systémů. Na obrázku č. 2 je vidět robot s polohovací postřikovou hlavou proti pesticidům. Lidský operátor na dálku na základě kamerového snímku v uživatelském rozhraní zobrazeném na obrázku č. 3 označí cíle pro postřik. [10, s. 3–5]

Ve výsledku je možné vidět, že z pohledu spolupráce mezi člověkem a robotem se výzkum a praktické aplikace zaměřují na dvě hlavní oblasti. Zaprvé to je zlepšování senzorických systémů v oblasti strojového vidění doplněním verifikace a korekcí robotického operátora. A za druhé jsou roboti využíváni jako asistenti pro zmírnění míry lidské manuální práce a snižování zátěže při namáhavých a nebezpečných úkolech. Míra dělby práce mezi strojem a člověkem silně závisí na povaze daného úkolu a potřebnou efektivitou, účinností a přesností. [10, s. 3–5]



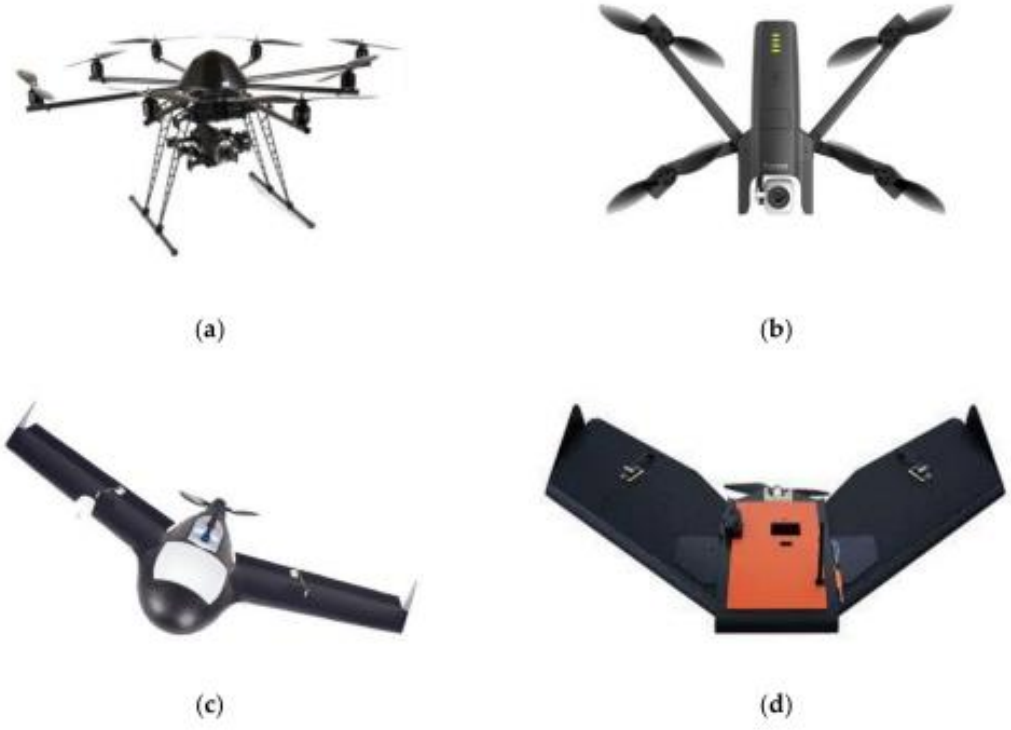
Obrázek 2: Postřikovací robot s kamerovým systémem [11]

Obrázek 3: Obrazový výstup s označenými cíli [11]

Bezpilotní vzdušné letouny

S využitím bezpilotních vzdušných letounů UAV (Unmanned Aerial Vehicles) se můžeme setkat při různých úkolech spojených s mapováním, vzdáleným snímkováním, monitorováním nebo kontrolou škůdců. Použití pouze jediného UAV pro tyto úkoly přináší určitá omezení zejména co se týče časové efektivity a omezené kapacity baterie. Typy architektury jednotlivých dronů využívaných v zemědělství jsou dost různorodé v závislosti na konkrétní aplikaci. Na obrázku č. 4, který zobrazuje čtyři příklady prakticky využívaných dronů, můžeme vidět rozdíly v počtu a umístění rotorů, umístění senzorů a typech křídel (rotační/pevná). Jedním z řešení je využití týmu spolupracujících UAV, kdy je úkol rozdělen mezi různá zařízení, která mohou koordinovat svůj pohyb a rozdělení plochy a tím zátěže. Takový přístup určitě snižuje dobu provádění úkolu, ale také může snížit spotřebu energie na úrovni jednotlivých UAV. Vzhledem k povaze zemědělských prací se výzkum v této oblasti zaměřuje na algoritmy pro řízení formace a pokrytí oblasti. Jedna z praktických aplikací se zabývá zaměřováním oblasti hejnem UAV, kde řídicí systém rozdělí oblast na čtverce o velikosti respektující charakteristiky použitých kamer. Každý letoun se poté snaží najít nezmapovaný čtverec na základě vzdálenosti od tohoto čtverce. Bezpilotní vzdušné letouny nejsou většinou pověřeny fyzickými úkoly na poli jako je setí, sadba nebo sklizeň. Místo toho jsou vybaveny kamerami a senzory pro monitorování daných parametrů, detekci škůdců nebo mapování.

Nasazení více jednotek na stejné území má za cíl dosáhnout rychlejšího pokrytí. Efektivita spolupráce a pokrytí oblasti poté závisí na nastavení a typu kooperačních algoritmů. Tyto algoritmy musí mimo jiné zohlednit vzdálenosti k cílovým čtvercům, pozice ostatních dronů ve formaci pro zamezení srážek a jednotlivé kapacity baterií. Vzhledem k neustále se rozšiřujícímu spektru technologických nástrojů bude budoucí výzkum směřovat k rozšíření možností využití UAV díky zlepšení jejich schopností ve vnímání, kapacitě a autonomnímu dobíjení baterií a manipulačních dovednostech. [10, s. 5–7]



Obrázek 4: Komerční UAV pro zemědělské účely. a) 8–rotorový MK Okto XL2, b) 4 rotorový Parrot Anafi, c) Gatewing X100, d) Tuffwing Mapper [12, s. 7]

Bezpilotní pozemní vozidla UGV

Tato vozidla se na rozdíl od monitorovacích funkcí UAV zabývají hlavně fyzickou prací na zemi. Stejně jako v předchozí kapitole se spolupráce v tomto případě pozemních robotů využívá pro urychlení a zefektivnění práce. Jedním z praktických příkladů je využití spolupráce pozemních vozidel při sklizni citrusových plodů. V této aplikaci je hlavní zaměření na hierarchické přiřazování úkolů a následné plánování trajektorií. Na jedné straně máme úroveň spolupráce, kde dochází k optimalizaci formace strojů a na druhé straně individuální plánovaní jednotlivých trajektorií. Skupina robotů využívá přístupu master-slave, kde vedoucí robot definuje pohyb celé skupiny a podřízená vozidla si na základě naplánované trasy celé skupiny navrhnou vlastní trajektorii. Na základě vzdálenosti od hlavního vedoucího si jednotliví podřízení počítají chybu od naplánované trajektorie a s využitím pozorovatele tuto chybu minimalizují. Hlavní výhodou tohoto přístupu je nízká výpočetní náročnost i při zahrnutí složitých dynamik strojů. [10, s. 8]

Projekt s názvem *MARS (Mobile Agricultural Robot Swarm)* se zabývá autonomními zemědělskými operacemi realizovanými koordinovanou skupinou robotů (obrázek č. 5). Klíčovým prvkem konceptu MARS je nízká individuální inteligence, což znamená, že každý robot je vybaven pouze minimálním množstvím senzorové technologie, aby byl dosažen nízkonákladový a energeticky úsporný systém. Jednotlivé komponenty tohoto systému jsou:

- roboti – váha do 50 kg, vybaveni osevní jednotkou a senzorem a upřesněným senzorem RTK-GNSS (korekce signálů z GPS, GLONASS s pomocí pozemní stanice pro přesné určení polohy v reálném čase),
- centrální logistická jednotka – transport/uschování robotů, zásobárna energie a osiva, pozemní stanice pro korekci GNSS signálů,
- OptiVison – centrální entita pro řízení shluku robotů, zajištění vzájemné komunikace, plánování, přiřazování úkolů,
- Cloud + UI – cloudové úložiště informací, uživatelské rozhraní pro počáteční nastavení a následné monitorování s možností intervence.

Centrální entita řídicí celý systém se skládá ze dvou komponentů – plánovací a řídicí. Celý proces je inicializovaný požadavkem od farmáře v uživatelském rozhraní na osetí pole. Tento požadavek obsahuje mapu GPS souřadnic pole s označenými statickými překážkami, osevní vzor (čtverec, řada, šestiúhelník), hustotu osevu, GPS pozice centrální logistické jednotky a vstupní brány a počet nasazených robotů. V první části algoritmu se vymezí okrajová část pole definovaná specifikovanou vzdáleností. V případě statických překážek uprostřed pole dojde k rozdělení na regiony a v jednotlivých regionech se celá plocha pokryje osevním vzorem (na obrázku č. 5 je osevním vzorem řada). Dalším krokem je přiřazení jednotlivých regionů konkrétním robotům ve snaze o rovnoměrné rozložení práce mezi všechny stroje. Poslední plánovací operací je určení vhodných bodů ve vhodných časech pro opuštění úkonu a doplnění osiva a energie u centrální logistické jednotky. Kontrolní část centrální entity poté monitoruje jednotlivé roboty a zajišťuje jejich komunikaci s cloudovým úložištěm. V případě odhalení nefunkčního zařízení ho řídicí jednotka začne brát jako statickou překážku a dojde k přeplánování. Úkol poškozeného robota poté převezme robot nejbližší, který obdrží nové instrukce k doplnění energie a osiva v závislosti na přidané práci. Tento přístup poskytuje velmi přesné zeměpisné údaje o zasetých rostlinách umožňující využití automatizace i v procesech hnojení a ochrany a následné sklizně. [14]

Obrázek 5: Spolupráce UGV na poli (vlevo), robotická jednotka MARS (vpravo) [13]

UGV jsou v porovnání s UAV využívány hlavně pro fyzické operace jako je setí, postřikování a sklizeň. Každý ze dvou zde uvedených přístupů – řídicí robot s jeho následovníky a řízení systému centrální entitou – má své výhody a nevýhody v závislosti na konkrétní aplikaci a podmínkách. Vhodným využitím spolupráce více robotů lze zvýšit výnos na jednotku plochy na farmě, snížit potřebné vstupy a enviromentální dopad (poškození půdy a využití pesticidů) a tím snížit finanční i energetické náklady na produkci.

Hybridní UAV/UGV představují spolupráci dvou výše zmíněných kategorií – pozemních a vzdušných autonomních dronů. Použití více UAV má výhody jako je velká oblast pokrytí a rychlost. Mezi jejich nevýhody patří nepřesnost pozemního měření, často nedosahující potřebných centimetrových požadavků a krátká doba letu související s nízkou kapacitou baterie. Na druhou stranu použití více UGV poskytuje možnost přesnějších pozemních měření a možnost vykonání fyzické akce. Praktické aplikace tohoto druhu spolupráce využívají silných stránek jednoho druhu autonomních dronů a kompenzují slabší stránky využití druhého typu. V rámci projektu Evropské unie *RHEA (Robot fleets for Highly Effective Agriculture)* byl vyvinut systém založený na spolupráci vzdušných a pozemních robotů pro mapování a následný postřik pole. Cílem bylo zásadně snížit množství herbicidů a hnojiva pro snížení kontaminace životního prostředí. Flotila robotů je řízená manažerem, který má moduly pro plánování, kontrolu, dohled a zpracování naměřených dat.

Vzdušná flotila (obrázek č. 6) byla v tomto projektu tvořena dvěma šesti-rotorovými drony s dobou doletu kolem 40 minut. Drony byly vybaveny dvěma kamerami ve viditelném spektru a v blízkosti infračerveného spektra a zařízením pro telemetrii a indikátorem stavu baterie. Jejich úkol byl na základě zadaného listu bodu proletět nad oblastí a pořídit fotografie.

Obrázek 6: RHEA – AR200 drony [15]

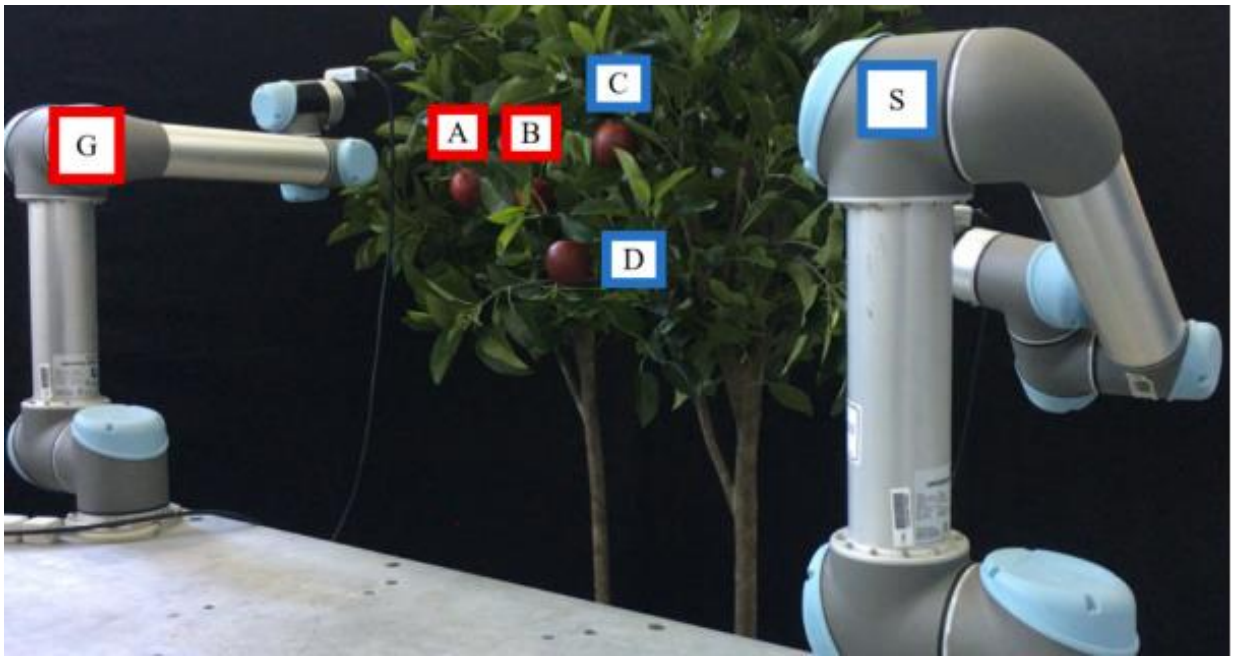
Pozemní flotilu (obrázek č. 7) tvořily tři traktory vybavené postřikovacím zařízením a sensory. Pro přesné určení polohy byly umístěny senzory poskytující GPS souřadnice zpřesněné korekcí z RTK antén. Dvě RTK antény také umožňují určit směr pohybu s přesností na desetinu stupně. Na vrchu traktoru byla namontovaná kamera ve viditelném spektru s vysokým rozlišením, která umožňovala detekci rostlin, překážek a linek na poli v reálném čase. Kvůli vysokým teplotám a proměnlivému počasí v letním období byla doplněna chlazením a krytem proti dešti. V přední části byl nainstalovaný lidar pro detekci objektů s lehkým náklonem pro detekci malých rostlin. Po stranách byly umístěny tyče pro samotný postřik a vzadu měl traktor dvě nádrže – jedna na vodu a druhá s herbicidy. Všechny senzory byly připojené do palubního počítače. [15]

Obrázek 7: RHEA – pozemní jednotka na postřik [15]

Obecně by UGV měly v těchto systémech poskytovat základní logistiku, přesná pozemní měření, informace ze senzorů pro vykonání zadaných zemědělských činností a v případě potřeby interakci s lidským činitelem. Častým konceptem je umístění nabíjecí a přistávací stanice na vrch pozemního vozidla, čímž dojde k značné úspoře energie u UAV. V budoucnosti by tato spolupráce mohla zvládnout úkoly typu – vzdušné mapování z dronů, tvorba elektronických map polí, monitorování nasazené techniky, posouzení potřeby aplikace hnojiv, odhalení oblastí napadených znečištěním nebo nemocí a v neposlední řadě samotný bezpilotní vzdušný postřik. [16]

Víceramenné systémy obsahují spolupráci hlavně při činnostech jako je sběr ovoce mezi několika operujícími rameny simultánně. Koncept tohoto kooperativního provozu zemědělských robotů lze rozšířit i na spolupráci mezi rameny typicky umístěnými na stejném robotovi. Hlavní výhodou tohoto přístupu je zvýšení efektivity a snížení doby vykonávání zemědělského úkonu. Kromě sklízení ovoce po vlastní ose je možná v mnoha případech i spolupráce ramen pro sklízení jednoho kusu ovoce například v případě zákrytu. Zároveň lze tyto řídicí principy použít také na robotická ramena na různých základnách.
[10, s. 12]

Kooperace dvou robotických ramen se šesti stupni volnosti a hloubkovými kamerami byla aplikována pro sběr jablek v sadu (obrázek č. 8). První rameno mělo za úkol jablko uchopit, zatímco druhé rameno se zaměřovalo na lokalizaci jablek skrytých mezi listy z pohledu prvního ramena. Pozice nalezeného jablka poté byla zanesena do grafu později využitého k výpočtu optimální trasy prvního ramena. [17]



Obrázek 8: Kooperace dvou ramen pro sběr jablek [17]

V řadě aplikací je stále v současné době vhodnější využít lidského pracovníka pro zejména koordinační a monitorovací úlohy. V rámci zmiňovaného druhu spolupráce robotů a lidí je důležité se zaměřit na volbu vhodného rozhraní tak, aby úkony mohly být obstarávány i pracovníky s minimálním technologickým školením. S rostoucí úrovní technologických dovedností robotů a umělé inteligence se dá předpokládat pokles potřebných lidských zásahů, až do doby, kdy procesy budou probíhat zcela autonomně. Monitorovací úkony v praktických zemědělských aplikacích mají stále jako největší překážku nízké energetické kapacity baterií, takže autonomní roboty/drony jsou omezeny pouze na krátké inspekční/mapovací úkoly.

1.1.3 Internet věcí – IoT (Internet of Things)

Internet věcí představuje velmi slibnou rodinu technologií, která má potenciál nabídnout mnoho řešení pro modernizaci zemědělství. Poskytuje prostředí integrující různé technologie (software) a zařízení (hardware), jako jsou bezdrátové telekomunikační technologie, senzory, RFID (radiofrekvenční identifikace) a další. IoT se člení do tří vrstev – senzorická, sítová a aplikační. [18, s. 1–2]

Senzorická vrstva je tvořena jednotlivými senzory získávajícími požadovaná data. Setkáváme se zde s technologiemi jako *NFC (Near Field Communication), RFID (Radio Frequency Identification)*, nebo už dříve zmiňovanými *WSN (Wireless Sensor Networks)*. NFC je bezdrátová technologie umožňující komunikaci mezi dvěma zařízeními, kde jedno je aktivní člen – generuje elektromagnetické pole – a druhé je člen pasivní. Při vstupu pasivního členu do elektromagnetického pole aktivního členu může probíhat mezi těmito zařízeními krátkodobá komunikace, nejčastěji se s touto technologií můžeme setkat při bezkontaktní platbě kartou na terminálu. RFID využívají štítky obsahující data ve formě elektronického produktového kódu. Čtečky RFID štítky „spouští“ a manipulují s uloženými daty například za účelem identifikace objektu nebo sledování. [18, s. 3]

Síťová vrstva poskytuje prostředí pro interakci mezi uzly bezdrátových senzorů a fyzickými objekty. Zajišťuje tím vzájemnou komunikaci ve formě předávání dat potřebnou pro uložení nebo další analýzu. Pro komunikaci jsou využity bezdrátové standarty usnadňující síťování jednotlivých zařízení. Mezi komunikační protokoly řadíme technologie např. ZigBee, Bluetooth Low Energy, LoRaWAN nebo Wi-Fi. [18, s. 4]

Aplikační vrstva je třetí vrstva zajišťující praktickou realizaci a řešení problémů s ní spojených. Hlavními vlastnostmi jsou jednoznačná identifikace připojených zařízení, spolehlivost, trvalost a škálovatelnost. Vize IoT spočívá v umožnění existence velkého počtu zařízení s velkou rozmanitostí technických specifikací (formát, napájení, výpočetní výkon, ...) v jedné interní síti. Middleware vrstva umístěná mezi zařízeními a aplikační vrstvou umožňuje jedné aplikaci běžet zároveň na více platformách a operačních systémech. Zároveň využívá přístup architektury orientované na služby – *SOA (Service Oriented Architecture)*, který snižuje čas strávený na aktualizacích aplikace a díky nezávislosti na technologiích lze opakovaně využívat hardware. [18, s. 4]

Aplikace založené na IoT se v zemědělství používají například pro monitorování plodin a chovu zvířat, sledování zemědělských strojů, zavlažování a kontrolu kvality vody, monitorování půdy, počasí a nemocí napadajících jednotlivé druhy rostlin. Mezi hlavní sektory využití patří:

- monitorování,
- zemědělská technika,
- skleníková produkce.

Monitorování je důležité pro správu různých faktorů v zemědělské činnosti, protože nám pomáhá porozumět vlivům proměnných na výslednou produkci. Mezi data získaná monitorováním mohou v závislosti na konkrétní aplikaci patřit: množství srážek, relativní a absolutní vlhkost vzduchu, atmosférická teplota, vlhkost a složení půdy atd. Zemědělská technika je v konceptu Zemědělství 4.0 vybavená velkým množstvím senzorů sbírajících data, která se odesílají na příslušná místa v síti. Na základě získaných dat jsou příslušné algoritmy nebo samotní zemědělci schopni v reálném čase korigovat jízdu vozidla, plánovat trasu pro hnojení a zavlažování. Efektivní využití těchto technologií má schopnost značně zvýšit produktivitu a snížit ztráty způsobené nesprávným obhospodařováním. Při skleníkové produkci má farmář mnohem větší kontrolu nad prostředím a je schopen více a jednodušeji regulovat faktory ovlivňující růst a kvalitu plodin. Využití sítě senzorů propojené v koncepci IoT má možnost značně snížit spotřebovanou energii a množství potřebného hnojiva a vody. Výzvy v této oblasti jsou závislé na konkrétní aplikaci, ale na obecné rovině patří mezi hlavní problémy bezpečnost a soukromí dat, spolehlivost, lokalizace zařízení, rušení signálů a optimalizace využití zemědělských zdrojů. [19, s. 6–7]

Cloud Computing

Cloud computing je způsob poskytování konfigurovatelných sdílených IT zdrojů, jako jsou výpočetní kapacity, síťové služby nebo úložiště. Tyto zdroje jsou k dispozici na vyžádání a mohou být snadno navýšeny nebo jinak upraveny pode potřeb uživatele.

Cloud je k dispozici pomocí tří servisních modelů – software jako servis *(SaaS),* platforma jako servis *(PaaS)* a infrastruktura jako servis *(IaaS).* V modelu *SaaS* mají uživatelé možnost přistupovat k různým softwarům a databázím uloženým na cloudu bez instalace na vlastní zařízení. Za správu verzí programů je zodpovědný správce. Model *PaaS* poskytuje přístup k vývojové platformě s nainstalovaným operačním systémem za účelem vývoje vlastních aplikací. Poslední model *IaaS* umožňuje uživatelům například využívat virtuálních strojů, úložišť nebo služeb VPN ve vysokorychlostní síti. [20, s. 1–2]

S praktickým využitím cloudových technologií se můžeme setkat ve všech oblastech od vědecké a technické až po obchodní a společenskou. V zemědělství se setkáváme hned s řadou praktických aplikací:

- uložiště velkého množství informací – databáze informací o plodinách, počasí, zemědělských postupů a konkrétních zkušeností,
- levný způsob přístupu k IT zdrojům – výhodná možnost redukce nákladů spojených s nákupem zdrojů využitím pronájmu,
- Cloud Agro Systém – počítačový systém k monitorování informací týkajících se zemědělství,
- automatizace záznamů o zemědělských pozemcích – analýza půdy, historie produkce,
- nástroje pro sběr dat – ze senzorů a bezdrátových sítí.

Použití této technologie usnadňuje zemědělcům mimo jiné správu a monitorování zemědělských aktivit. Výsledkem pak může být zvýšená produkce, snadnější marketing a efektivně uložená a zpracovaná data pro následné rozhodovací procesy. [20, s. 3–4]

1.1.4 Datová analýza

Výše zmiňované moderní technologie se zabývají shromažďováním, sdílením a uchováváním informací a dat. V řadě praktických aplikací v dnešní době množství získaných informací značně převyšuje zpracovatelské schopnosti běžných metod. V tomto odvětví se setkáváme s dvěma technologiemi typickými pro zpracování dat v 4.0 konceptech – big data a strojové učení spojené s umělou inteligencí.

Big data, jak název napovídá, reprezentují velké a komplexní soubory dat z dané oblasti, které jsou těžce analyzovatelné běžnými metodami. Rozlišujeme u nich několik typických charakteristik:

- objem – velikost dat sesbíraných pro analýzu,
- rychlost – časové okno, ve kterém jsou data ještě užitečná a relevantní,
- rozmanitost – rozdíly v klíčových vlastnostech dat (typ, časový interval, rozlišení, …),
- pravdivost – míra přesnosti a spolehlivosti dat,
- zhodnocení – identifikace využitelných dat.

Zdrojem dat v případě zemědělství mohou být data ze senzorů (pozemních, vzdálených), meteorologických stanic, historických sad dat, systémů doplňujících data o jejich přesnou lokalizaci nebo z lidmi shromážděných informačních zdrojů (mobilní aplikace, průzkumy). [21] Data je poté dále možno analyzovat pomocí několika metod:

Regresní analýza zkoumá vztah mezi jednou nebo několika nezávislými proměnnými a jednou závislou proměnnou. Poskytuje schopnost zjistit důležitost vztahu mezi závislými a nezávislými proměnlivými a posoudit vliv několika nezávislých proměnných na jednu závislou. Podle počtu nezávislých proměnných, tvaru regresní křivky nebo typu závislých proměnných rozlišujeme různé druhy regrese. Praktickým případem může být závislost výnosu sklizně na datech o teplotě, pH půdy, obsahu dusíku, fosforu a draslíku v půdě, spadlých srážkách nebo potřebné vodě. [21]

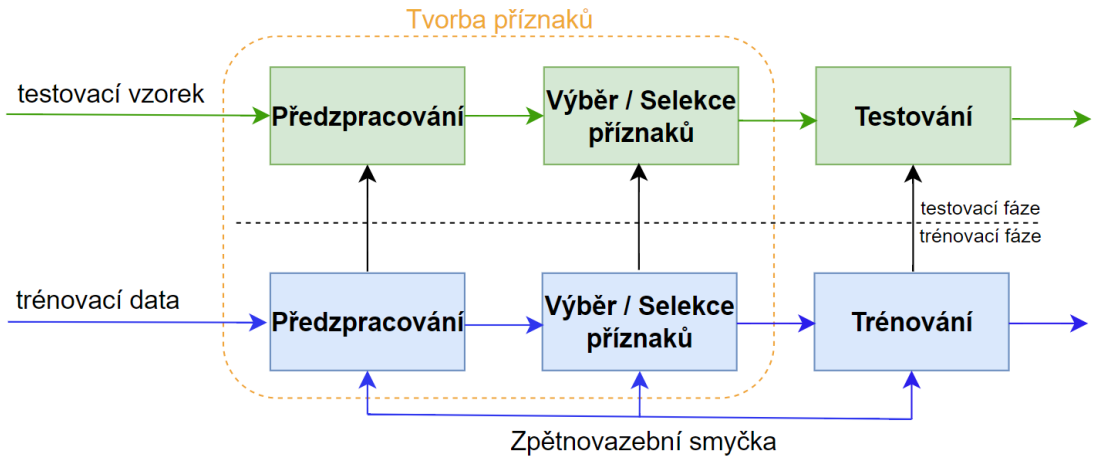
Shlukování (Clustering) reprezentuje metody umožňující spojování různých zemědělských objektů za účelem optimalizace a identifikace jednotlivých zón. Podle požadovaného výstupu a dodaných vstupů rozlišujeme velký počet shlukovacích algoritmů (K-means, Fuzzy C-means,

mean-shift, …). Přístup obecného shlukovacího algoritmu můžeme rozdělit do jednotlivých kroků: nalezení parametrů jednotlivých objektů → získání optimálního množství shluků → identifikace center shluků → zisk výsledných parametrů shluků (tvar, velikost).

Při využití shlukovacího algoritmu založeného na hustotě můžeme získat z dodaných datových sad o průměrné teplotě, srážkách a vlastnostech půdy oblasti s podobnými charakteristikami v závislosti na dodaných vstupech. [21]

Klasifikace je metoda zaměřená na kategorizaci objektů v závislosti na jejich vlastnostech – prediktorech. Spočívá ve volbě a trénování modelu, který je po natrénování schopen kategorizovat nové objekty. Využité metody klasifikace mohou být od rozhodovacích stromů přes SVM až po komplexnější přístupy hlubokého učení. V praxi jsme poté po dostatečném natrénování schopni například bezpečně rozeznat jednotlivé rostliny na poli nebo dokonce jednotlivé druhy rostlin. [21]

Strojové učení je metoda, jejíž cílem je optimalizovat výkon úkolu využitím příkladů nebo minulých zkušeností. Operuje ve dvou fázích znázorněných na obrázku č. 9 – trénovací a testovací. V trénovací fázi se algoritmus učí vykonávat zadané úkoly na trénovacích datech až do doby, dokud není splněna matematicky vyjádřitelná podmínka. Vstupní data prochází předzpracováním, kde se často čistí od nekonzistentních a chybějících částí, integrují z více zdrojů a transformují pomocí normalizace a diskretizace. Cílem extrakce/výběru příznaků je identifikovat nebo vytvořit podmnožinu příznaků, ve které bude model během trénovací fáze implementován. V testovací fázi je model vystaven testovacím datům, která ještě nikdy neviděl, a následně rozhodne buď metodou klasifikace nebo regrese o existenci příznaků v jednotlivých příkladech. Podoborem strojového učení je tzv. hluboké učení, které využívá alternativní architekturu přesouvající proces konvertování dat na příznaky příslušnému systému učení. V důsledku toho blok extrakce/výběru příznaků není potřeba, což vede k plně vyškolenému systému, který z nezpracovaných dat dosáhne požadovaného výstupu. [22]



Obrázek 9: Proces strojového učení [upraveno podle [22]]

Typy učení se dělí na:

- S učitelem – stroj hledá cestu, jak se dostat od známého vstupu ke známému výstupu.
- Bez učitele – trénovací data jsou bez označení, stroj si sám generuje strukturu na základě vstupů.
- Zpětnovazební učení – model preferuje rozhodnutí, která vedou k maximalizaci nějaké formy odměny.

Využití umělé inteligence a strojového učení se v dnešní době rozšiřuje do více odvětví, kde v řadě aplikací hraje klíčovou roli. V konceptu Zemědělství 4.0 se s algoritmy, které ho využívají, můžeme setkat v souvislosti se správou plodin (predikce sklizně, detekce rostlin a jejich druhů, plevele a nemocí), efektivním využíváním vodních zdrojů a správou půdy nebo hospodářských zvířat.

1.1.5 Systémy pro podporu rozhodování

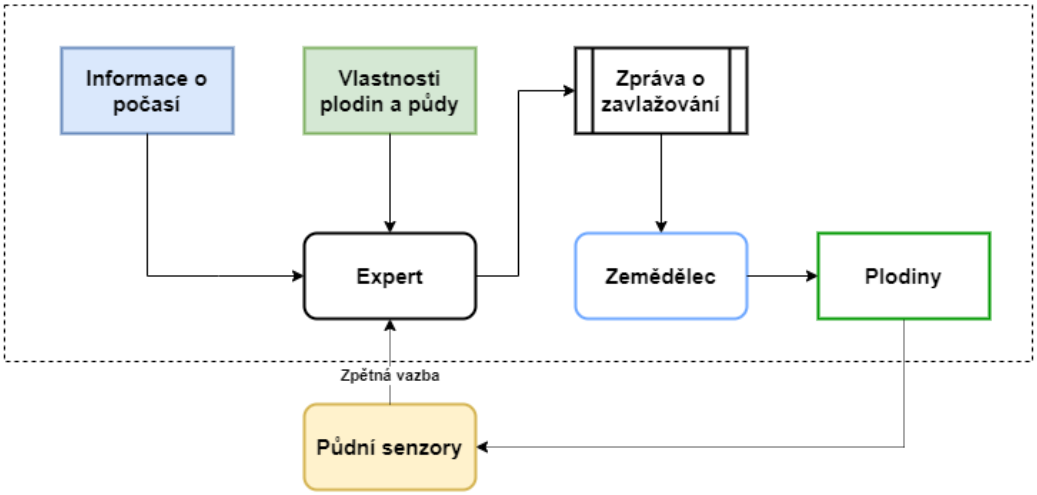
Systémy podpory rozhodování *DSS (Decision Support Systems)* se v průběhu historie vyvíjely s cílem pomoci uživatelům systematicky řešit některé zásadní problémy z příslušných oborů. Tyto systémy značně závisí na odborném posouzení návrháře od správné definice problému přes výběr relevantních dat až po volbu přístupů k tvorbě výsledného řešení. Jejich cílem je prezentovat koncovému uživateli analyzovaná data a předat seznam možných navržených postupů. I přesto, že jsou pro správu zemědělských struktur velmi užitečné, je jejich využívání omezeno několika zásadními problémy. Jelikož jsou tyto systémy stále v rané fázi, co se praktického využití týče, farmáři často nemají dost zkušeností s jejich používáním. Samotné navržené aplikace mohou být často neintuitivní pro běžného uživatele, zaměřené pouze na řešení podproblémů bez uvažování klíčových faktorů v širším hledisku. [2] [23]

Citovaná literatura rozpoznává čtyři hlavní sektory využití podpůrných systémů pro zemědělské účely:

- plánování mise,
- hospodaření s vodními zdroji,
- adaptace na změnu klimatu,
- kontrola plýtvání potravinami.

Plánování mise se zaměřuje hlavně na rozdělení úloh mezi jednotlivá zařízení a plánování jejich cest v terénu. Efektivní dělba práce je zásadní pro zvýšení produkce, úsporu spotřebované energie a celkový čas trvání konkrétních úkonů. Fungování projektu MARS, které se touto tématikou také zabývá bylo nastíněno v přechozí části práce.

Hospodaření s vodními zdroji je klíčové pro zavlažovací systémy hlavně v sušších oblastech s omezenou vodní zásobou. Úsudky a rozhodnutí lidských expertů dosahují dobrých výsledků, ale nejsou škálovatelné a dostupné pro každé pole/farmu. Z toho důvodu a pro značné zrychlení zdlouhavé analýzy byl navržen systém podpory pro zavlažování využívající metod strojového učení. Vývojový diagram na obrázku č. 10 představuje princip fungování navrženého systému. Informace o počasí z meteorologické stanice v okolí spolu s informacemi o vlastnostech plodin a půdy jsou analyzovány expertem. Výstupem je zpráva obsahující údaje o potřebném množství vody na daný časový usek, která slouží zemědělci jako reference pro samotné zavlažování. Navržený podpůrný systém by měl nahradit po důkladném natrénování na historických datech roli lidského experta. [24]



Obrázek 10: Vývojový diagram systému pro řízení zavlažování [upraveno podle [24]]

Za účelem **adaptace na změny klimatu** zejména v subtropických oblastech byl představen DSS integrující model simulace plodin s geografickým informačním systémem. Ten obsahuje data z databáze WorldClim obsahující historická i aktuální data o počasí. Tato data jsou poskytnuta modelu simulujícímu růst plodin CROPGRO, který má k dispozici odpovídající půdní profily pro jednotlivé lokace. Porovnáním výstupů z modelů o potenciální sklizni v jednotlivých lokacích poskytne zemědělci představu o možném výnosu na základě zvolených parametrů modelu. [25]

Pro **kontrolu** a snižování **plýtvání potravinami** se podpůrných rozhodovacích systémů využívá například pro optimalizaci dodavatelského řetězce. Na základě informací o požadavcích spotřebitelů, zemědělské produkce a stavu skladů s potravinami jsou nalezeny optimální dodavatelské cesty, které umožňují doručovat produkty na dané místo v nejkratším čase. Zároveň mohou informace od spotřebitelů sloužit jako indikátor pro zemědělce pro přizpůsobení plánů pěstování. [2]

Je důležité připomenout, že využití DSS má často nízkou úroveň autonomie. Systémy podle zadaných parametrů data zpracují a poskytnou uživateli přehled a návrh řešení, koncové rozhodnutí je ale v této fázi uplatnění stále jen na koncovém uživateli.

2 Fúze senzorů

Cílem datové fúze senzorů je kombinace informací z více zdrojů za účelem zisku pohledů nedosažitelných použitím pouze jednoho ze senzorů. V předešlých kapitolách byly senzory popsány jako zařízení schopná rozlišit specifické atributy a změny v prostředí a následně poskytnout zpětnou vazbu systému. Většina senzorů neměří rovnou požadovaný fenomén, ale výsledný signál se generuje v jednotlivých krocích, s čímž může být v konečném důsledku spojeno zkreslení výsledné informace. Tyto statické vlastnosti jako přesnost, rozlišení a citlivost se často dají dobře nastavit a kalibrovat před samotným měřením a fúzí. Větším problémem jsou dynamické vlastnosti více závislé na vstupech jako rychlost a frekvence odezvy, čas ustálení a zpoždění. Některé chyby jsou způsobené náhodným šumem, což vyžaduje dodatečné zpracování signálu za účelem redukce chyby. Systematická chyba korelovaná s časem se dá v případě, že ji známe, odstranit pomocí definovaného filtru. [26, s. 14–23] [27]

Výhod využití dat získaných fúzí senzorů v porovnání s daty z individuálních senzorů je hned několik. První samozřejmá výhoda vychází z rozšíření dat z jednoho snímače o data z druhého jiného/identického snímače, čímž se nám rozrostou celková data k analýze a výsledek by poté měl být přesnější. S tím souvisí i zvýšená jistota naměřených dat, zvlášť pokud se dostaneme dvěma nezávislými cestami k podobným nebo stejným výsledkům. Je tak mnohem snadnější odstranit nebo rovnou redukovat chybu způsobenou poškozeným nebo špatně kalibrovaným senzorem. Další výhoda se často projeví u fúze dat z odlišných senzorů o odlišných informacích. Doplnění dat z radaru o kamerové snímky nám poskytne novou řadu korelovaných informací a jsme schopni například identifikovat poznávací značku vozidla, které v určitém místě v obci překročilo povolenou rychlost. Posledním zde uvedeným příkladem je zisk polohy zařízení při známých vzdálenostech od jednotlivých čidel metodou triangulace. Konkrétnější výhody jsou úzce spjaté s praktickou aplikací, nejde tedy obecně určit například optimální počet senzorů, vždy je potřeba vycházet z konkrétního příkladu. [26, s. 23–24] [27]

Můžeme se ale setkat i s řadou možných obtíží a problémů: [27]

- Registrace – individuální senzory operují ve vlastním referenčním rámci a je třeba najít společný rámec, adresovat kalibrační chyby a data sjednotit.
- Nejistota – existence různých formátů dat může vytvořit šum, nejasnosti nebo výsledky, které si protiřečí.
- Nekompletní, nekonzistentní, chybná data – často se dá vyřešit doměřením chybějících dat nebo opětovným měřením v případě konfliktních výsledků.
- Korespondence a asociace – zabývá se problematikou posouzení, zdali data reprezentují stejný objekt a dále určení, z jakého senzoru informace pocházejí
- Zrnitost – často nejednotná míra detailu jednotlivých měření.
- Časové měřítko – řešení sjednocení informací s rozdílnými frekvencemi, šíření zpoždění v systému, registrace a ukládání historických dat s odpovídajícím časovým štítkem.

2.1 Obecný model fúze

Zobecněnou fúzi dat aplikovatelnou na více odvětví popisuje model JDL na obrázku č. 11. Vstupní data prochází blokem datové fúze propojeného s databází odkud vystupují na nějaké uživatelské rozhraní. Celý blok je rozdělený na pět podúrovní.

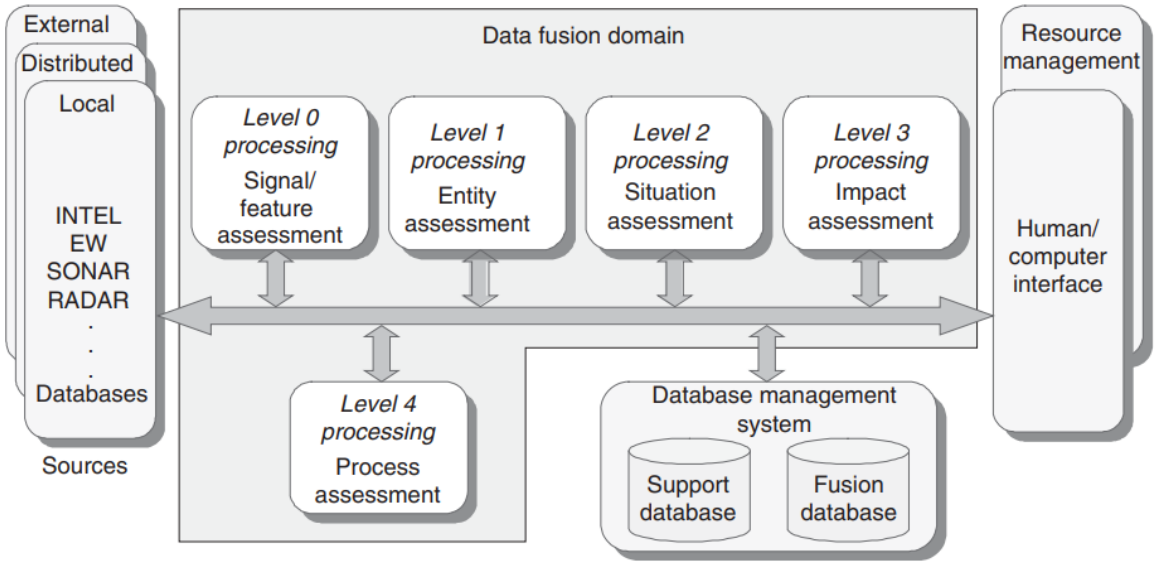
Podúroveň 0: Zde dochází k prvotnímu zpracování obdržených signálů jako například extrakce příznaků z obrázků nebo analogových signálů či odhad parametrů v elektromagnetických nebo akustických datech. Hlavní důraz je kladen na strukturu měřených dat, a ne na jejich původ a význam.

Podúroveň 1: V této podúrovni jsou identifikovány jednotlivé entity/objekty, jejich klasifikace, kinematika, nebo atributy klíčové pro danou aplikaci.

Podúroveň 2: Identifikované objekty jsou zde dále zkoumány z hlediska dané situace. Zaměření je zde na vyvozování jednotlivých vztahů uvnitř a mezi entitami, klasifikace těchto vztahů a jejich implikace.

Podúroveň 3: Zde dochází k predikci dopadu aktuálních charakteristik na celkový stav v budoucnosti.

Podúroveň 4: Tato podúroveň se stará o monitorování celého procesu fúze s cílem zlepšit výkon v reálném čase.



Obrázek 11: Revidovaný model fúze dat JDL [26, s. 52]

Existují tři hlavní přístupy ke vzájemnému kombinování dat ze senzorů.

Vysokoúrovňová fúze HLF (High-Level Fusion)

V tomto přístupu vykoná každý senzor zadaný úkol (detekce nebo sledování objektu) samostatně a tyto výsledky jsou poté spojovány dohromady. Patří k nejvyužívanějším přístupům díky nízké komplexitě. Bohužel může poskytovat nedostačující informace, jelikož pokud dojde například k překryvu několika objektů, jsou související hodnoty nedůvěryhodné a nejsou v celkové fúzi uvažovány. Příkladem využití HLF může být například propojení radarových měření a lidarového mraku s následným využitím nelineárního Kalmanova filtru pro detekci objektů a sledování.

Nízkoúrovňová fúze LLF (Low-Level Fusion)

Na rozdíl od HLF se data fúzují dohromady na nejnižší možné úrovni hned z nezpracovaných nebo minimálně zpracovaných dat. Tímto se zachovají všechny informace, což může potenciálně zlepšit detekci. V praxi tento přístup čelí řadě výzev v neposlední řadě s implementací. Pro přesné sloučení je potřeba správná kalibrace senzorů pro statický i dynamický provoz.

Fúze střední úrovně MLF (Mid-Level Fusion)

Jedná se o abstraktní úroveň, ve které dochází k datové fúzi na úrovni jednotlivých příznaků. Nalezené příznaky ze surových dat (například informace o barvě obrazu nebo shlucích v lidarových nebo radarových měřeních) jsou použity k následným procesům jako je rozpoznávání nebo klasifikace. Často nelze dosáhnout dobré přesnosti kvůli limitovanému vnímaní prostředí a ztrátě informací poskytujících kontext. [29, s. 24]

2.2 Rozdělení základních metod

Publikovaná literatura na téma fúze senzorů se často liší v rozdělení algoritmů a metod, které se jí zabývají. V této práci bude popsána klasifikace z odborného článku rozdělující metody fúze senzorů pro využití v sektoru samořiditelných aut na tradiční přístupy a přístupy využívající hlubokého učení. [28]

2.2.1 Metody tradičního přístupu

Tyto algoritmy jsou založeny na teorii nejistoty a zpracování nedokonalých dat. Tabulka č. 1 porovnává metody z této kategorie, uvádí jejich stručnou charakteristiku, jejich výhody a nevýhody, oblast aplikace a úroveň fúze. V této části budou déle uvedeny a stručně vysvětleny jednotlivé algoritmy a jejich principy z uvedených skupin metod.

Tabulka 1: Porovnání jednotlivých metod tradičního přístupu k fúzi senzorů [upraveno podle [28, s. 8–10]]

Skupina metod	Charakteristika	Výhody	Nevýhody	Oblast aplikace	Úroveň fúze
Statistické	Využití pro doplnění dat pomocí statistického modelu	Dokážou zpracovat neznáme korelace, tolerantní	Limitované na lineární estimátory, vysoká komplexnost výpočtů	Odhad	Nízká
Pravděpodobnostní	Založeno na pravděpodobnostním rozložení senzorických informací	Zpracování nejistoty v poskytnutých informacích, využito i pro nelineární systémy	Vyžaduje předchozí znalosti o modelu systému a datech	Odhad, Klasifikace	Nízká až Střední
Založené na znalostech	Využití přístupů k výpočetní inteligenci na bázi lidské inteligence	Dokážou zpracovat nejistoty a nepřesnosti, využití pro komplexní nelineární systémy	Založené na expertních znalostech a způsobu jejich získávání	Klasifikace, Rozhodování	Střední až Vysoká
Důkazního uvažování	Založeno na Dempsterově kombinačním algoritmu pro implementaci modelu	Poskytnuté informace doplněny o stupně nejistoty, identifikace konfliktních situací, modelování komplexních předpokladů	Vysoká komplexnost výpočtů, vyžaduje předpoklad důkazu	Rozhodování	Vysoká
Analýzy intervalu	Sdílení provozního prostoru v intervalech, problém splnění omezení.	Zaručení integrity, schopnost zpracování komplexních nelineárních systémů	Diskretizace provozního prostoru, vysoká komplexita	Odhad	Nízká

Statistické metody

Kalmanův filtr

První ze skupiny algoritmů využívajících modelu je nejpopulárnější technika pro odhad – Kalmanův filtr. Principem je využití explicitního statistického modelu popisujícího průběh veličiny x v čase a modelu pozorovatele.

$$x(k+1) = A(k)x(k) + B(k)u(k) + w(k) \quad (1)$$

$$z(k) = H(k)x(k) + v(k) \quad (2)$$

Rovnice č. 1 popisuje odhad stavu diskrétního procesu $x(k+1)$ z matice přechodu $A(k)$, matice výstupů $B(k)$, vektoru vstupů $u(k)$ a $w(k)$ reprezentujícího stavový šum procesu s Gaussovským rozdělením. Druhá rovnice popisuje vektor pozorování $z(k)$ určený pomocí matice měření $H(k)$ a šumu měření $v(k)$. Dále jsou definovány matice kovariance $Q(k)$ a $R(k)$. Na základě těchto rovnic postupuje filtr ve dvou etapách – predikce a aktualizace. Ve fázi predikce dojde k odhadu stavové veličiny $\hat{x}(k)$ a dopočtu kovariance $P(k)$.

$$\hat{x}(k) = A(k)x(k-1) + B(k)u(k) \quad (3)$$

$$P(k) = A(k)P(k-1)A^T(k) + Q(k) \quad (4)$$

Následuje fáze aktualizace, ve které se na základě aktuálního měření $z(k)$ dopočte korekce odhadu stavové veličiny $\hat{x}(k)$.

$$\hat{x}(k) = \hat{x}(k) + K(k)[z(k) - H(k)\hat{x}(k)] \quad (5)$$

$K(k)$ zde představuje zesílení Kalmanova filtru určené ze vztahu:

$$K(k) = P(k)H^T(k)\ [H(k)P(k)H^T(k) + R(k)] \quad (6)$$

Aktualizovaná kovariance $P(k)$ je určena ze vztahu, ve kterém I značí jednotkovou matici:

$$P(k) = [I - K(k)H(k)]P(k) \quad (7)$$

Kalmanův filtr je využíván hlavně pro nízkoúrovňovou fúzi dat. V případě existence lineárního modelu a šumu modelovaného Gaussovým šumem poskytuje Kalmanův filtr optimální statistický odhad.

Modifikací KF je takzvaný *rozšířený Kalmanův filtr EKF (Extended Kalman Filter*), který pracuje s nelineárními modely. Jeho hlavní nevýhodou jsou náročné výpočty Jakobiánů. V případech, kdy je to možné je tendence modely linearizovat, je ale potřeba brát zřetel na vzniklé odchylky.

Pro aplikaci v distribuovaných systémech, kde je třeba kombinovat informace z různých zdrojů se využívá *distribuovaný Kalmanův filtr DKF (Distributed Kalman Filter).* Jeho hlavní myšlenkou je kombinace jednotlivých lokálních stavů pro určení stavu celkového, což značně snižuje nutnou vzájemnou komunikaci mezi jednotlivými senzory. V těchto distribuovaných systémech je kladen velký důraz na přesnou synchronizaci hodin. [27] [31, s. 12–13]

Průnik kovariancí (Covariance Intersection)

Tato metoda popisuje přístup k fúzi kombinující dva odhady s neznámými vzájemnými korelacemi. Mějme dvě informace A a B, jejichž fúzí máme získat výstup C. Problém je nedostatečná znalost informací, A i B by mohly být buďto dvě rozdílná měření nebo A by mohl

být odhad z modelu systému a B informace ze senzoru. Při známých A a B můžeme vyjádřit páry průměru a kovariance $\{\hat{a}, A\}$ a $\{\hat{b}, B\}$, kde po aplikace CI můžeme určit pár $\{\hat{c}, C\}$ jako:

$$C^{-1} = \omega A^{-1} + (1 - \omega)B^{-1} \tag{8}$$

$$\hat{c} = C(\omega A^{-1}\hat{a} + (1 - \omega)B^{-1}\hat{b}) \tag{9}$$

Hodnoty označené stříškou reprezentují odhadovaný průměr, C kombinovanou kovarianční matici a ω koeficient, který dává informaci o konzistenci výsledného odhadu. [30]

Pravděpodobnostní metody

K-nejbližších sousedů KNN (K-Nearest Neighbours)

Metoda nejbližších sousedů NN je jednou z nejjednodušších technik zabývajících se datovou asociací hledáním shluků podobných hodnot. Výsledné tvořené shluky závisí na zvolené metrice a příslušném nastavení prahu pro tuto metodu. Jedná se o jednoduchý algoritmus, který je schopen najít příslušné řešení za krátkou dobu, bohužel v chaotických nepřehledných prostředích zanesených šumem nedosahuje moc dobrých výsledků.

Jeho nejznámější modifikací je metoda K-Means, která rozděluje hodnoty datové sady do K rozdílných shluků. Jedná se o iterativní algoritmus hledající nejlepší rozmístění těžišť shluků operující v následujících krocích:

1) Získání vstupních dat a počtu K požadovaných shluků.
2) Náhodné zvolení těžišť každého shluku.
3) Přiřazení jednotlivých bodů z dat příslušným shlukům.
4) Posunutí těžiště shluku do geometrického středu shluku
5) Dokud algoritmus nekonverguje, návrat zpět do kroku 3

Jedná se o běžně využívaný algoritmus, bohužel je s jeho užitím spojeno několik nevýhod. Zaprvé ne ve všech případech najde optimální řešení pro střed shluku a je potřeba nastavovat prahový parametr. Nejpraktičtějším řešením tohoto problému je nechat algoritmus běžet několikrát a jako výsledek vybrat ten s nejmenší variancí. Druhou nevýhodou je nutná znalost počtu shluků, co chceme identifikovat ještě před spuštěním. Možné řešení v řadě aplikací je začít s malými hodnotami k a postupně se propracovat k požadovanému rozdělení. A poslední zde uvedenou nevýhodou je buďto neuvažování kovariance datové sady nebo, že je sada normalizovaná, což se dá odstranit vynásobením dat maticí inverzní ke kovarianční matici. [31, s. 6–7]

Bayesovské metody

Fúze informací založená na Bayesovském odvození poskytuje způsob kombinace důkazů podle pravidel teorie pravděpodobnosti. Nejistota je reprezentovaná pravděpodobnostními termíny, které nabírají hodnot v intervalu $< 0;\ 1 >$, kde 1 znamená absolutní přesvědčení a 0 přesný opak. Bayesovský odhad je definován následovně:

$$P(Y \mid X) = \frac{P(Y \mid X)\ P(Y)}{P(X)} \tag{10}$$

Pozdější (posteriorní) pravděpodobnost $P(Y \mid X)$ reprezentuje důvěru k hypotéze Y vůči známé informaci X. Tuto pravděpodobnost můžeme získat vynásobením předchozí (apriorní) pravděpodobnosti $P(Y)$ pravděpodobností, že X platí za předpokladu, že Y je pravdivé

$P(Y \mid X)$. Hodnota $P(X)$ hraje roli normalizační konstanty pro zachování intervalu $< 0;\ 1 >$. Hlavní nevýhodou je nutná předchozí znalost pravděpodobností $P(X)$ a $P(Y \mid X)$. Dalším problémem spojeným s touto metodou je přílišná komplexnost v případě existence většího počtu hypotéz a událostí závisejících na určitých podmínkách. [31, s. 15]

Metody založené na znalostech

Umělé neuronové sítě ANN (Artificial Neural Networks)

Neuronové sítě jsou matematické modely složené z nelineárních výpočetních elementů – neuronů pracujících paralelně v různých topologiích. V porovnání se standartní lineární a nelineární analýzou jsou mnohem účinnější a přizpůsobitelnější. Neuronové vrstvy se dají učit různými způsoby. Na základě trénovacích dat a učícího algoritmu si příslušně nastavují jednotlivé vnitřní váhy. Po dosažení maximální dovolené chyby si pamatují své nastavení a mohou být nasazeny na reálná data. Hlavní výzvou je volba správné topologie pro daný problém. [27, s. 4]

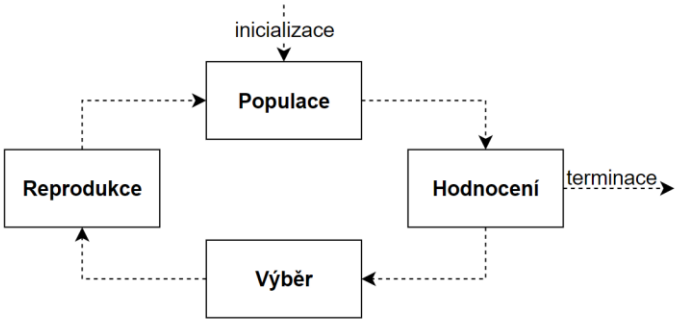
Fuzzy logika

Jedná se o disciplínu, která je v poslední době čím dál častěji využívána ve vysokoúrovňové fúzi. Hlavní výhodu přináší svým systémem pro reprezentaci neurčitosti, kde každému výroku přiřazuje jako míru pravdivosti hodnotu v intervalu $< 0;\ 1 >$. Rozvíjí tak binární systém, ve kterém hodnota 1 odpovídá pravdě a 0 nepravdě a umožňuje existenci nových stupňů pravdivosti někde „mezi“ často vyjádřených charakteristickým slovem. Vysoký stupeň přesnosti a jistoty, kterou tato logika poskytuje, je bohužel spojený s často vysokou komplexností výpočtů. Ve většině praktických aplikací fúze senzorů se můžeme setkat s kombinací tradičních a na umělé inteligenci založených metod ve snaze využít z každé její výhodnou část. [27, s. 4–5]

Genetické algoritmy

Jedná se o hledací algoritmy založené na darwinovské teorii přírodního výběru a přežití nejsilnějších jedinců. Jsou schopny efektivně poskytnout řešení komplexních problémů, o kterých nevíme moc podrobností. Od svého vzniku v sedmdesátých letech našly své využití v řadě vědních oborů od robotiky, optimalizace funkcí, zpracování obrazů nebo identifikaci a řízení systémů. První funkční operací genetického algoritmu je formulace problému. Je nutné stanovit všechna možná řešení problému konvertovaná do binárních řetězců a definovat funkci, kterou má genetický algoritmus optimalizovat buď dosažením maxima nebo minima. Dále musíme doplnit problém o specifické znalosti například o jednotlivých proměnných nebo krajních omezeních jejich rozsahů a zahrnout genetické operátory jako křížení a mutaci.

Po formulaci problému přichází na řadu genetické vyhledávání zobrazené na obrázku č. 12. Ve fázi *inicializace* jsou náhodně generovaná řešení rozprostřená po celém rozsahu možných řešení pro vytvoření prvotní populace. Velikost populace závisí na podstatě problému, ale řádově se pohybujeme ve stovkách až tisících řešení. Každý jedinec populace je poté *hodnocen* na základě definované fitness funkce. Z každé populace jsou poté *vybráni* jedinci na základě hodnocení pro založení nové generace. Ve fázi r*eprodukce* jsou vybraní jedinci modifikování genetickými operátory jako křížení (ze dvou rodičů vznikne jeden jedinec) nebo mutace (před křížením jsou rodiče náhodně pozměněni). Nová populace je poté tvořena nejlepšími jedinci a nejhorší jedinci jsou nahrazeni kříženými potomky. Celý proces se opakuje po mnoho generací, dokud není nalezeno přijatelné řešení nebo dojde k překročení nadefinovaného počtu generací. [32]



Obrázek 12: Proces genetického vyhledávání [upraveno podle [32]]

Metody důkazního uvažování

Dempster – Shaferovo odvození

Tato metoda má za úkol generalizovat výše zmíněné Bayesovské odvození. Poskytuje také způsob reprezentace nekompletních znalostí, aktualizace přesvědčení a umožňuje nám explicitně vyjádřit nejistotu určitého děje. Přiřazuje každému elementu rámce rozlišování – hypotéze H pravděpodobnost váhové funkce v intervalu $< 0;\ 1 >$. Dempsterův kombinační algoritmus se tedy využívá právě k aktualizaci a kombinaci těchto váhových funkcí (mass functions). Vstupní váhové funkce popisující pravděpodobnosti hypotéz jsou kombinovány za účelem zisku výsledné váhové funkce, která popisuje odhad situace na základě všech dostupných hypotéz. Zásadním rozdílem proti Bayesovskému odvození je absence potřeba apriorních znalostí pravděpodobností. [31, s. 15]

Metody analýzy intervalu

Jedná se o skupinu metod založených na konceptech analýzy intervalu a šíření omezení. Hlavní myšlenkou je reprezentace čísel intervaly, které obsahují jejich hodnoty, čímž zahrneme i jejich nejistotu nebo chybu. Kombinací těchto konceptů vznikají metody s názvem šíření intervalového omezení *ICP (Interval Constraint Propagation).* Jejich cílem je iterativním procesem aktualizovat zadané intervalové reprezentace s ohledem na systém omezení, limitující veličiny pomocí matematických vztahů. Jsou užitečné v případech, kdy je nutné modelovat nejistotu a zaznamenávat její šíření na základě omezení a podmínek. [33]

2.2.2 Metody hlubokého učení

Metody využívající hlubokého učení pro fúzi senzorů jsou v této práci uvedeny pro její kompletnost. Z algoritmů hlubokého učení zde bude jen stručně naznačen princip fungování konvolučních neuronových sítí.

Tabulka 2: Porovnání jednotlivých metod hlubokého učení pro fúzi senzorů [upraveno podle [28, s. 11]]

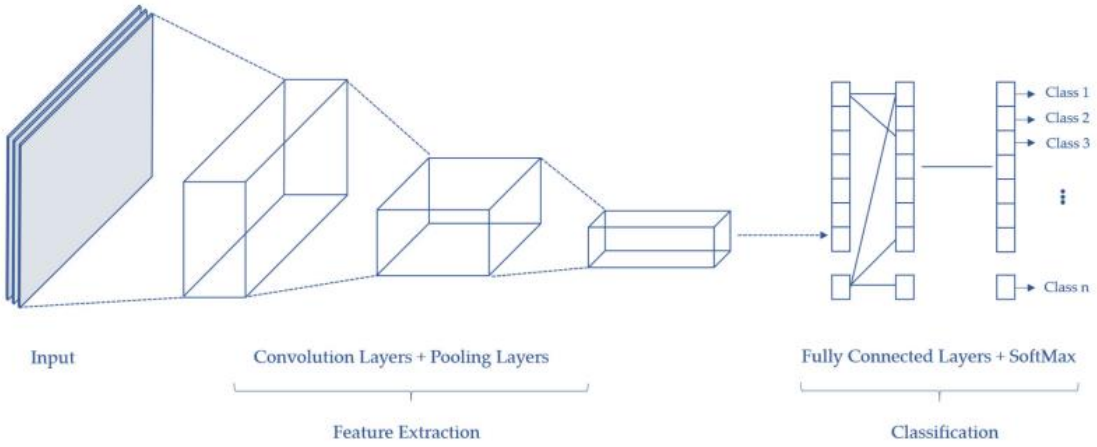
Algoritmus hlubokého učení	Stručný popis	Příklady aplikace
Konvoluční neuronová síť (CNN)	Sítě s dopředným přenosem s konvolučními a sdružovacími vrstvami. Velmi účinné pro hledání závislostí mezi jednotlivými pixely ve zpracování obrazu.	Počítačové vidění, rozpoznání řeči
Rekurentní neuronová síť (RNN)	Zpětnovazební sítě, využívající přechozích výstupů k predikci nových vzorků dat. Pracují s daty jako sekvencemi, buď vstupů nebo výstupů.	Popis obrázků, předpověď dat, zpracování přirozeného jazyka
Síť hluboké víry (DBN)	Vícevrstvý energetický model s viditelnou vstupní vrstvou a několika skrytými vrstvami. Přiřazuje pravděpodobnostní hodnoty parametrům svého modelu.	Spolupracující filtrování, rozpoznání ručně psaných znaků, hlasové rozpoznání

Autoenkodéry (AE)	Skupina neuronových sítí, které často využívají způsobů učení bez učitele. Jsou složeny z enkodérů a dekodérů a dají se trénovat pomocí minimalizací rozdílů mezi vstupem a výstupem	Redukce dimenze, získávání obrázků, odstranění šumů z dat

Konvoluční neuronové sítě

Jak bylo popsáno v tabulce č. 2, jedná se o sítě složené z konvolučních a sdružovacích vrstev organizovaných ve všech třech dimenzích dosahující výborných výsledků při zpracovaní obrazů. Schéma na obrázku č. 13 představuje bližší pohled na rozdílné vrstvy zpracování dat. Vstupní vrstva běžně obsahuje data o vstupujícím obrazu. Následující sekce je složená z konvolučních sítí detekujících a extrahujících důležité příznaky a sdružovacích (pooling) vrstev umístěných mezi sítě konvoluční za účelem snížení výpočetní náročnosti redukcí některých prostorových informací za současného zachování důležitých vlastností. Vybrané příznaky jsou poté klasifikovány v plně propojených vrstvách jednotlivých neuronů (fully connected layers). Výstupní vrstva obsahuje výsledek ve formě pravděpodobnosti klasifikace.

[28, s. 12–16]



Obrázek 13: Schéma CNN [28, s. 12]

2.3 Oblasti aplikace fúze senzorů

Využití vzájemného spojování informací z více zdrojů je obecný postup uplatitelný na libovolné odvětví. Jedná se o rychle se měnící odvětví, ve kterém je pro nejaktuálnější informace potřeba sledovat nejnovější literaturu a postupy. Po automatizační a senzorické stránce můžeme ale jmenovat určité oblasti, ve kterých se už v současné době bez algoritmů spojujících tyto zdroje neobejdeme.

Internet věcí (IoT) byl v této práci už detailněji probírán. Jeho charakteristické velké množství dat sbíraných v reálném čase pro následnou analýzu je ideálním předpokladem pro využití datové fúze. Se stále přibývajícím počtem zařízení připojených k internetu se dá očekávat, že rozvoj v této oblasti poroste, například v automatizaci oborů spotřeby energie, elektrické sítě a chytrých domácností. [27]

Jednou z v dnešní době nejskloňovanějších oblastí pro uplatnění senzorických informací je automobilový průmysl. Moderní elektrické automobily například od firmy Tesla jsou vybavovány značným počtem senzorických zařízení vyžadujících vzájemnou komunikaci od GPS snímače, lidar senzorů nebo ultrazvuku. Využití v tomto odvětví je hlavně pro tvorbu adekvátní reprezentace okolního prostředí prostřednictvím 3D map, detekce objektů a jejich klasifikace pro umožnění autonomního řízení. Každá použitá technologie má své výhody, pro

rychlý zisk informací je vhodné využívat radarový senzor, pro úseky plné zatáček nebo městské oblasti je vhodnější využívat lidar s mnohem větším polem záběru. [27]

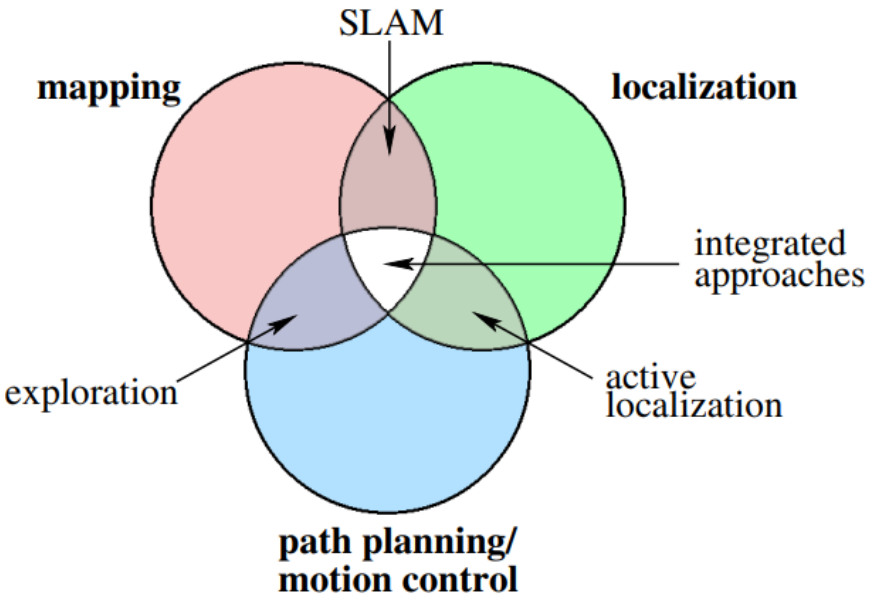
Problematika dronů a kvadrokoptér byla nastíněna u kooperativní robotiky v souvislostí s UAV. Navigační systém těchto vznášedel je většinou tvořen trojosým gyroskopem, akcelerometrem a magnetometrem doplněné o senzory GPS, ultrazvuku a tlakového výškoměru. Pokrok v oblasti autonomního letu umožní přístup do míst, kam se jinak lidsky navigované drony těžce dostávají. Důležitým aspektem je nepřerušená funkce systému i při chybném/chybějícím vstupu z nějakého senzoru. [27]

Další oblastí, kde dochází k fúzi sensorů ať už několika stejného typu nebo různých, jsou vojenské aplikace v souvislosti s námořní plavbou a využíváním elektromagnetických a sonarových vln pro detekci. Detekce je jeden z hlavních úkolů většiny vojenské technologie, jelikož dřívější objevení potencionální hrozby dává značnou taktickou výhodu. Do této kategorie spadají i protiletadlové senzory a zbraně nebo systémy protiraketové obrany. [26, s. 4–5]

Neposlední uplatnění nachází kombinace datových zdrojů v avizovaném počítačovém vidění. S výrazným prohloubením našeho pochopení procesů vnímaní značně roste i úroveň počítačového vidění a stává se nepostradatelným pomocníkem v mnoha aplikacích včetně medicíny, robotiky nebo jakékoliv statické detekce. [27]

3 Robotické mapování

Modelování prostředí je potřebné pro řadu robotických aplikací od autonomních aut po robotické vysavače. Obecný problém mapování se dá rozdělit na tři úkoly – *mapování*, *lokalizace* a *plánování trasy*. *Mapování* je problém integrace informací shromážděných ze senzorů robota do určité reprezentace, *lokalizace* je problém odhadování pozice robota v závislosti k mapě a plánování trasy se zabývá efektivním naváděním vozidla na požadovanou lokaci nebo po určité trajektorii. V reálných aplikacích se tyto problémy málokdy dají řešit separátně. Diagram na obrázku č. 14 zobrazuje tyto spolu související úkoly a vysvětluje i disciplíny vzniklé jejich vzájemným průnikem. Současná lokalizace a mapování se nazývá *SLAM (Simultaneous Localization and Mapping).* Tyto dva úkoly jdou ruku v ruce, jelikož pro dobrou lokalizaci je mít třeba dobrou mapu a obráceně. *Aktivní lokalizace* se snaží navigovat robota na místa v mapě s cílem zlepšení odhadu pozice na rozdíl od *průzkumu*, který využívá přesných informací o pozici a strategicky naviguje daným prostředím pro tvorbu mapy. Průnikem všech tří úkolů jsou tzv. *integrované přístupy* někdy nazývané jako *SPLAM (Simultaneous Planning, Localization and Mapping).* Výsledkem těchto přístupů je autonomně se pohybující robot současně vytvářející mapu prostředí. [34, s. 3–4]



Obrázek 14: Úkoly spojené s robotickým mapováním [34, s. 4]

3.1 Problémy a výzvy

Robotické mapování s sebou přináší řadu výzev, které autor mapovacího algoritmu nebo uživatel, který se snaží mapu vytvořit musí zohlednit. Robot může být v závislosti na aplikaci vybaven řadou senzorů jako jsou kamery, dálkoměry (lidar, sonar, radar), infračervená technologie, taktilní senzory, kompasy, akcelerometry a gyroskopy, GPS senzory atd. Každý z těchto senzorů má své vlastní parametry ovlivňující jeho přesnost a měření.

Nepřesnosti nazývané jako šum měření jsou jednou z klíčových výzev jakéhokoliv mapování, a to z důvodu, že reálný šum je často statisticky závislý, takže ho nelze eliminovat opakovaným měřením. V řadě měření jako například při zisku pozice vozidla z odometrie dochází ke kumulaci chyb, čím déle je stroj v pohybu. Na konci měření se tak můžeme dostat k velikým odchylkám. Druhá komplikace vychází z potřeby vysokého počtu znalostí k modelaci jedné entity. Dvojdimenzionální detailní plán místnosti vytvořený robotickým průzkumem může být složený klidně z tisíce bodů, kde každý bod reprezentuje pouhý odhad spojený s nepřesností

měření. Třetím problémem je datová asociace, což je určení, zda měření v jiných bodech v čase popisují ten stejný objekt. Dá se řešit obvykle využitím jedinečných identifikátorů nebo metodami sledování objektu v průběhu času. Dalším problémem je snaha mapovat v čase proměnné prostředí. Existují způsoby, jak se s dynamikou prostředí vypořádat, ale ve většině aplikací uvažujeme místo toho prostředí statické a jakékoliv časové změny jsou brány jako šum a patřičně odfiltrovány. Neposledním problémem je volba cesty při prozkoumávání a mapování prostoru, která může být o to komplikovanější, pokud se snažíme efektivně koordinovat skupinu robotů za účelem dělby práce pro urychlení procesu. Problémy spojené s robotickými mapami zde nastíněné se dají jednoduše shrnout do pěti bodů: [34, s. 5], [35, s. 5–6]

- navádění robota při autonomním průzkumu,
- vypořádání se se šumem v odhadu pozice a v pozorováních,
- interpretace senzorických dat a adresování nejistot v robotickém modelu,
- modelování změn v prostředí,
- efektivní vzájemná koordinace více robotů v prostředí.

3.2 SLAM (Simultaneous Localization and Mapping)

Jedná se o velmi důležitou a aktivně prozkoumávanou část robotiky, jelikož schopnost tvořit mapy a zorientovat se v prostoru je klíčovou pro pokrok autonomních mobilních robotů. Důležitým bodem, který nebyl zmíněn je historický vývoj k pravděpodobnostním přístupům v mapování. Většina nejmodernějších algoritmů v této oblasti operuje právě na bázi pravděpodobnosti z důvodu existence šumů měření. Cílem SLAMu pro jednoho robota je tedy posteriorně vypočítat pravděpodobnost mapy a trajektorie vzhledem k počáteční poloze robota a akčním a měřícím signálům. Matematicky se tato závislost dá vyjádřit jako maximalizace daného pravděpodobnostního vztahu, kde $x_{1:t}$ představuje sekvenci stavů za čas t, m reprezentaci prostředí formou mapy, $z_{1:t}$ sekvenci senzorických pozorování za čas t a $u_{1:t}$ sekvenci odometrických omezení za čas t.

$$p(x_{1:t}, m | z_{1:t}, u_{1:t}) \tag{11}$$

Stav robota je většinou vyjádřen nějakou kombinací pozice robota a jeho orientace. Mapa je silně závislá na typu využitého algoritmu, ale obecně je popsaná pomocí listu vytyčených orientačních bodů. Výše popsaný vztah reprezentuje takzvaný úplný SLAM. Jeho alternativou je SLAM uvažující pouze nejnovější stav robota, nazývaný filtrovaný/online SLAM, maximalizující totožný vztah, pouze s jednou hodnotou x:

$$p(x_t, m | z_{1:t}, u_{1:t}) \tag{12}$$

Tento přístup značně ulehčuje využití algoritmů typu Bayesův filtr, který pracuje opakováním funkce přechodu stavu a následnou aktualizací. Odometrie se pak využívá převážně v kroku šíření stavu a omezení měření pro následnou aktualizaci měření. Mapovací algoritmy se pak dále dají rozdělit podle využité technologie nebo podle mapy, kterou vytvářejí. [37, s. 3–4]

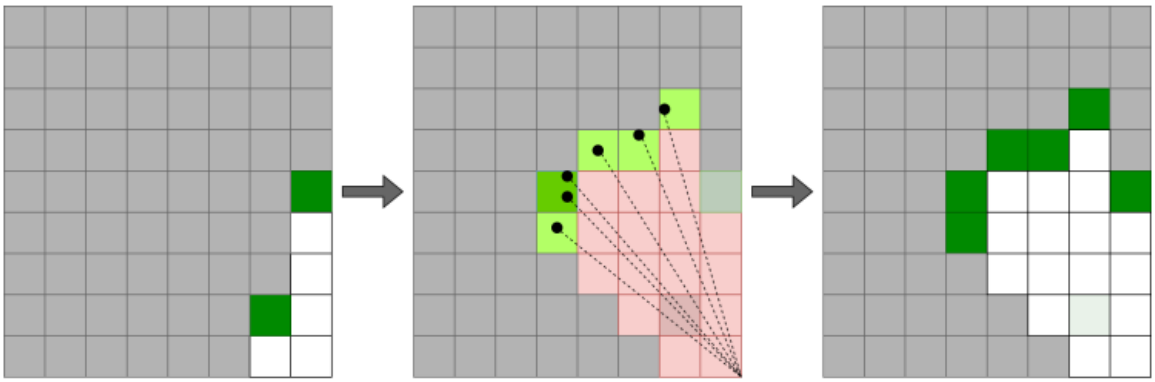
SLAM algoritmus se musí vypořádat se třemi kroky – senzorické informace, zpracování dat a reprezentace prostředí mapou. Jakmile robot zachytí data pomocí vhodných senzorů, přichází na řadu zpracování dat. Většinou se jedná o procesy filtrování, vyhlazování nebo využití umělé inteligence při zohlednění nejistot spojených s výpočty, algoritmy a modelováním. Tyto procesy jsou pak zakončeny příslušnou mapovou reprezentací. [36, s. 5]

3.3 Typy robotických map

Robotická reprezentace prostředí na sebe bere formu map, které jsou v této části rozděleny podle jejich typu a jednotlivé typy jsou stručně vysvětleny. Každý typ se zaměřuje na zobrazení jiných klíčových prvků jiným způsobem s často rozdílnými sektory využití. Vždy záleží na konkrétní aplikaci, která určuje, co má být v prostředí rozlišeno a identifikováno.

3.3.1 **Mřížkové mapy** (Grid maps)

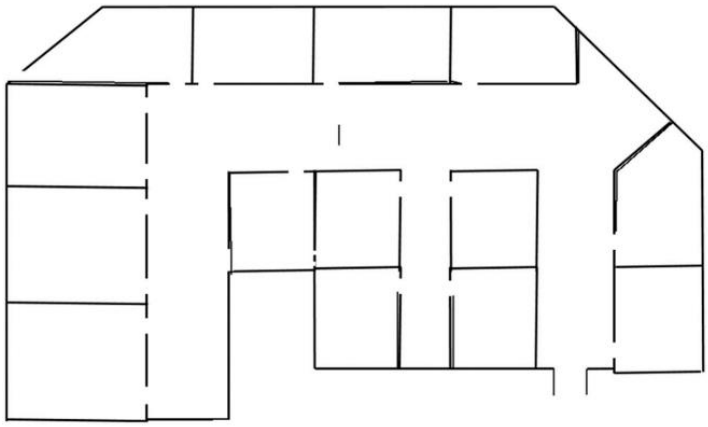
Jak z názvu vychází, prostředí je v tomto druhu map reprezentováno mřížkou. V původním návrhu autora této metody měly jednotlivé buňky pouze dva stavy – obsazená a neobsazená. Moderní mapování rozšířilo tuto koncepci, kde každá buňka obsahuje pravděpodobnost obsazenosti. Jedná se o populární přístup využívaný hlavně u tvorby 2D map, ale je možné ho modifikovat i o další dimenzi, i když se jedná o docela drahou záležitost. Obrázek č. 15 naznačuje tvorbu takovéto mapy pomocí sekvence měření nějakého dálkoměru. Bílá barva čtverců značí volný prostor, zelená barva značí objevené překážky (obsazené čtverce) a šedá mapa zobrazuje zatím nezmapovaný prostor. Jak můžeme vidět na prvním obrázku, prvotní měření objevilo obsazený čtverec, který se v následujícím měření ukázal jako volný. Výhodou této metody je právě možnost využití v dynamicky proměnném prostředí. Nevýhodné jsou v případě, kdy chceme dosáhnout požadovaného rozlišení → nutné značně zmenšit rozměry buněk. [36, s. 5–7]



Obrázek 15: Tvorba mřížkové mapy založené na obsazenosti [37, s. 10]

3.3.2 **Mapy příznaků** (Feature maps)

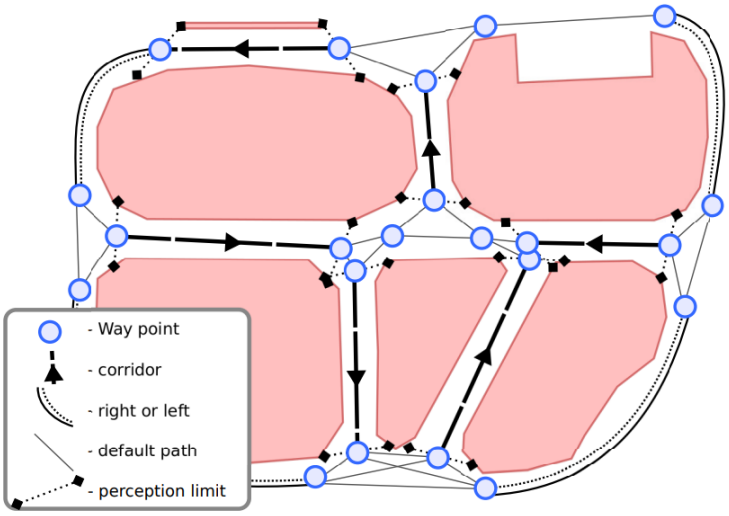
Tyto mapy reprezentují prostředí pomocí identifikovaných orientačních bodů. Jsou často velmi výhodné pro lokalizaci a prokazují schopnost efektivně pracovat s rostoucím objemem dat a úloh, což naznačuje dobrou schopnost škálování. Mapa na obrázku č. 16 zobrazuje půdorys podlaží, kde jsou jako jednotlivé identifikační body/příznaky vybrány zdi a jejich průchozí části. K identifikaci jednotlivých zdí musí být použito příslušných algoritmů pro extrakci a identifikaci těchto identifikačních bodů. Druhou nevýhodou je řešení jednoho z nejobtížnějších problémů v robotickém průzkumu, což je datová asociace z jednotlivých měření. [36, s. 5–7]



Obrázek 16: Mapa příznaků půdorysu podlaží [38, s. 9]

3.3.3 Topologické mapy

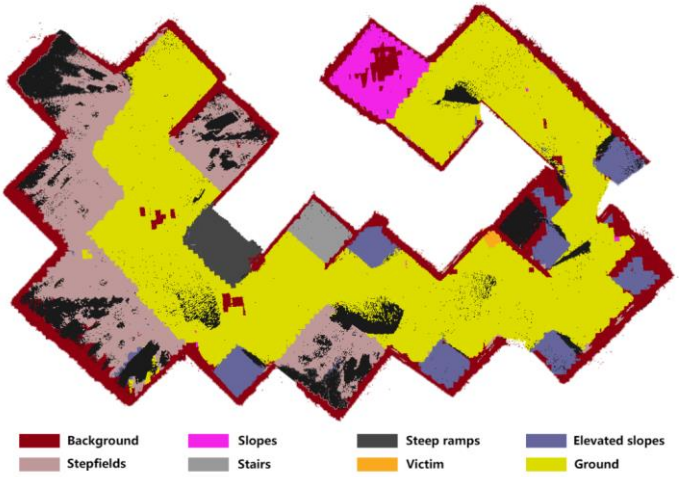
Zde je představen přístup k reprezentaci prostředí pomocí abstraktního modelu prostředí formou souhrnu propojených cest a uzlů (obrázek č. 17). Topologické mapy jsou nejčastěji využívány pro navigaci, plánování cesty a určení pozice. Vyžadují většinou nějakou formu zpracování, například odvození z mřížkových map založených na obsazenosti. Za nevýhodu by se dala považovat navigace pouze po vyznačených cestách mezi jednotlivými uzly. [36, s. 5–7]



Obrázek 17: Topologická mapa prostředí [39, s. 6]

3.3.4 Sémantické mapy

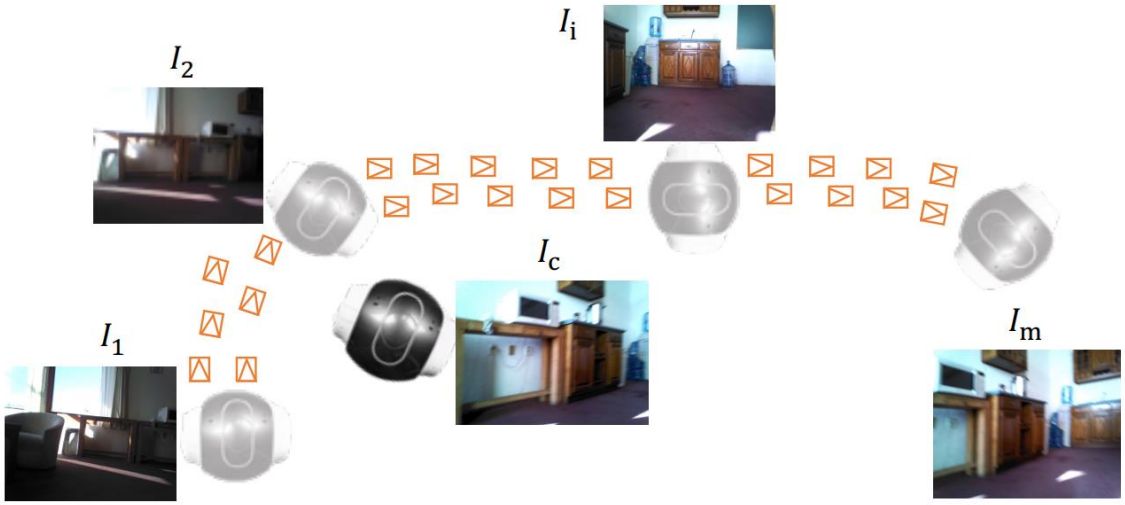
Tento druh map klade hlavní důraz na funkcionalitu a vztahy mezi jednotlivými objekty a prostředím. Jednotlivé objekty a oblasti jsou pojmenovány a označeny, často i odlišnou barvou pro lepší vizualizaci. Jsou důležité v reprezentaci prostředí pro vykonávání složitějších úkonů vyžadujících nějakou úroveň rozhodování. Vzhledově mohou být podobné předchozímu typu map, důležitým rozdílem je odlišení jednotlivých částí a objektů. Mapa na obrázku č. 18 zobrazuje půdorys patra budovy. Na rozdíl od pojetí sémantických map z jiných vědních oborů přiřazují sémantické mapy v oblasti robotiky jednotlivým částem prostoru význam. Různé kategorizace v prostoru jsou barevně odlišeny od ostatních. Nevýhodou je nutnost vývoje a trénování systému za účelem rozpoznávání objektů a jejich klasifikace. [36, s. 5–7]



Obrázek 18: Sémantická mapa s odlišenými místnostmi [40, s. 7]

3.3.5 **Mapy vzhledu** (Appearance maps)

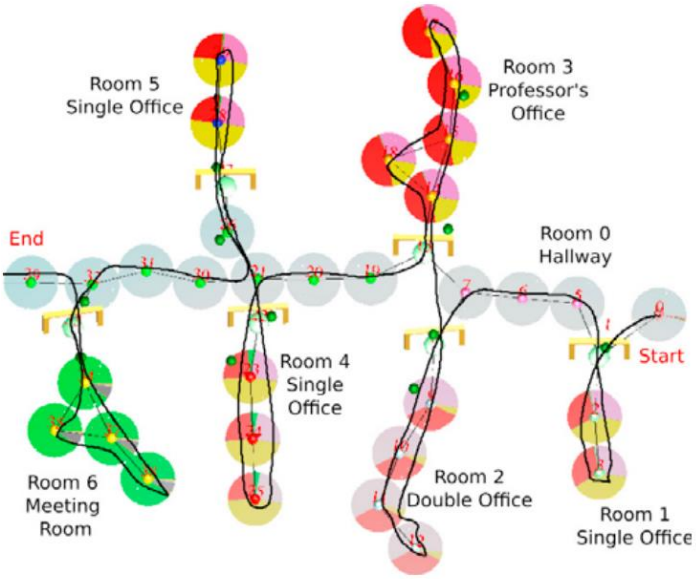
Mapy vzhledu kombinují jednotlivé pohledy s informacemi o místě, odkud byly pořízeny. Jednotlivé uzly v mapě jsou tradičně spojeny s nějakou formou obrazu v závislosti na použité vizuální technice. Jejich používání je velmi intuitivní a umožňuje si rychle vytvořit představu o mapovaném místě. Na rozdíl od přechozích map, obsahují kromě bodových nebo textových informací i informace o jednotlivých pixelech snímků, což značně zvyšuje požadovanou úložnou kapacitu. Obrázek č. 19 představuje tuto metodu reprezentace prostoru na konkrétní místnosti. Kromě oranžovou barvou vyznačené trajektorie obsahuje tato mapa i informace o natočení kamery v jednotlivých bodech a zajišťuje tak přehledné spojení mezi jednotlivými obrázky a prostorovými informacemi. [36, s. 5–7]



Obrázek 19: Příklad mapy vzhledu místnosti s vyznačenými pohledy [41, s. 3]

3.3.6 Hybridní mapy

Tyto mapy kombinují různé z výše zmiňovaných typů pro poskytnutí větší úrovně detailu a porozumění. Příkladem může být například doplnění topologické mapy o metrické údaje o vzdálenosti nebo situace zobrazená na obrázku č. 20. Zde je topologická mapa doplněná o sémantické údaje značně zvyšující celkovou přehlednost. Kombinace map vyžaduje značnou míru zpracování jednotlivých map, a hlavně vzájemnou koordinaci a sjednocení měřítka. Výhodou kromě poskytnutí většího množství informací najednou může být i identifikace špatně naměřených částí a případná oprava. [36, s. 5–7]



Obrázek 20: Hybridní kombinace topologické mapy a sémantických informací [42, s. 7]

3.4 Metody SLAMu s využitím laseru

Tato kapitola popisuje základní tři metody zpracovávání laserových měření za účelem současné lokalizace a mapování.

3.4.1 Částicový filtr

Celým názvem RB částicový filtr *RBPF (Rao-Blackwellized Particle Filter)* je jeden z prvních algoritmů používaných k SLAMu. Vyžaduje velké množství „částic“ pro dosažení dobrých mapovacích schopností, což s sebou přináší vysokou komplexnost výpočtů. Hlavním cílem je tedy optimalizace částicového filtru, tak aby došlo k značnému snížení počtu částic.

Převzorkování na základě důležitosti *SIR (Sampling Importance Resampling)* je algoritmus operující v následujících krocích.

1) Fáze predikce – generace velkého množství vzorků na základě funkce přechodu stavu (využití vážené sumy vzorků pro posteriorní odhad hustoty pravděpodobnosti).
2) Fáze korekce – přiřazení korespondujících vah důležitosti reprezentujících pravděpodobnost, že bude odhadnutá částice pozorována.
3) Fáze převzorkování – přerozdělení vzorků na základě rozdělení vah a využití přerozdělené sady částic v novém filtračním kole pro odhad nových částic s využitím funkce přechodu stavu.
4) Odhad mapy – vypočtení příslušného odhadu bodu mapy z trajektorie a pozorování vzorkování.

Po obdržení nových pozorování, SIR algoritmus zhodnotí váhy od začátku, což znamená že s časem značně roste výpočetní komplexita. Při využití převzorkování s nízkou variancí, nejsou částice generovány nezávisle na sobě, ale vždy se náhodně vygeneruje jen první částice a zbylé se generují v závislosti na první při respektování pravděpodobností odpovídajícím jednotlivým váhám.

Příkladem algoritmu založeném na částicovém filtru je například *Gmapping*, jehož modifikace vychází z vyzvání k převzorkování jen když počet částic klesne pod danou hranici, čímž se značně redukuje počet vzorků. [34, s. 7–11], [43, s. 1–3]

3.4.2 Porovnávání snímků (Scan matching)

Tento algoritmus využívaný pro hledání stejných struktur ve snímcích a jejich následné spojování úzce vychází z algoritmu iterativního nejbližšího bodu *ICP (Iterative Closest Point),* jenž má následující kroky.

1) Výběr bodu v mraku bodů z prvního měření.
2) Nalezení odpovídajícího bodu v následujícím měření a hledání vzájemné korespondence.
3) Vygenerování transformační matice ze známé korespondence a transformace zbytku bodů datové sady.
4) Zhodnocení kvality přiřazení pomocí kritéria průměrné vzdálenosti bodů.

$$\boldsymbol{dRMS}(\boldsymbol{P}*, \boldsymbol{Q}*) = \sqrt{\frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\|\boldsymbol{p_i} - \boldsymbol{q_j}\|)^{\mathbf{2}} < \boldsymbol{\sigma}, (\mathbf{1} \leq \boldsymbol{j} \leq \boldsymbol{m})} \quad (13)$$

Proměnné n a m jsou počty bodů, $\boldsymbol{q_j}$ je bod z původní sady dat, $\boldsymbol{p_i}$ je bod jemu odpovídající z následující sady dat a kritérium $\boldsymbol{\sigma}$ reprezentuje prahovou hodnotu minimální vzdálenosti sady dat. Pokud nedojde ke splnění kritéria, algoritmus se vrací zpět do kroku 2 a proces se opakuje.

Klíčové pro správné přiřazování odpovídajících si objektů jsou přesně známé rozdíly pozic mezi aktuálním a počátečním bodem. Existují tři způsoby přiřazování:

- bod k bodu – nejpomalejší, pomalá konvergence,
- bod k příznaku – identifikace příznaků v původním měření – shluky, linie, rohy a následné přiřazování bodů ke známým strukturám,
- příznak k příznaku – redukce stovek bodů na několik příznaků.

Algoritmus je ovlivňován řadou faktorů a při nevhodných podmínkách může místo správné registrace vyhodnotit optimální řešení pouze lokálně. Při nízkém poměru překrytí dvou datových sad nemusí kritérium správně identifikovat příslušný posun díky využití nejmenších čtverců. Problém může způsobit i příliš velké vzájemné natočení dvou datových sad nebo proměnlivá vzdálenost ve skenech. Dalším z mapovacích algoritmů na základě porovnávání snímků je například tzv. *Hector SLAM*. [43, s. 2–3] [44, s. 2–4]

3.4.3 Optimalizace grafů

Tato metoda vyjadřuje obecný optimalizační problém pomocí grafu. Prvním krokem tohoto algoritmu při použití pro mapování je konstrukce grafu, kde se jednotlivé souřadnice pozice v časovém úseku vynesou do grafu a propojí se spojnicemi. Druhým krokem je poté optimalizace grafu, snažící se například vyhlazovat jednotlivé hrany, nebo uzavřít smyčku. Uzavírání smyčky je jedním z často využívaných principů v případě, kdy měření v praxi probíhá v uzavřené smyčce a výsledná data díky kumulaci malých odchylek jsou od konečné pozice vychýlena. Příkladem využití metody optimalizace grafů je například algoritmus *Karto SLAM*. [43, s. 2–3]

3.5 SLAM Algoritmy

Současná lokalizace a mapování využívá různých principů závislých na dostupné technologii a výpočetní kapacitě. Tato sekce má za úkol poskytnout čtenáři přehled jednotlivých algoritmů rozdělených podle klíčových kritérií a propojit je s výše uvedenými principy. Existuje řada

dělení těchto metod, například podle typu vytvořené mapy nebo podle využité technologie. V této práci budou z hlediska zpracování dat budou popsány *filtrovací SLAMy*, *vyhlazovací SLAMy* a jako poslední bude zmíněno mapování z obrazových dat – *vizuální SLAM*.

3.5.1 Filtrovací SLAM

Tyto metody využívají při zpracování dat filtrů odvozených od Kalmanova nebo Bayesovského přístupu, jako *EKF-SLAM (Extended Kalman Filter SLAM), PF-SLAM (Particle Filter SLAM)* nebo *EIF-SLAM (Extended Information SLAM)*. Poslední příklad filtru využívajícího rozšířené informace, ve stručnosti pracuje s bayesovskou reprezentací informace pomocí pravděpodobnostního rozdělení. *EKF-SLAM* a *EIF-SLAM* spolu s několika dalšími můžeme řadit do skupiny algoritmů poskytujících řešení na základě vektorů obsahujících odhady mapy a pozice. Typicky jsou v těchto algoritmech procesy odhadu stavu a datové asociace řešeny separátně pomocí heuristiky nebo pokročilých filtrovacích metod, jako například metoda založená na společné pravděpodobnosti *JPDA (Joint Probability Data Association).*

Druhý typ algoritmů produkuje řešení založená na koncepci náhodných konečných množin. V množinových SLAM algoritmech jsou klíčové problémy jako počet příznaků a datová asociace začleněny do Bayesovského filtru, takže není nutné je řešit separátně. Tyto algoritmy jsou díky tomu mnohem lépe schopny zvládat nepřesná měření a chybnou nebo chybějící detekci. [36, s. 9]

3.5.2 Vyhlazovací SLAM (Smoothing SLAM)

Metody vyhlazení jsou založeny na výše popsaném problému optimalizace grafů. Snaží se o odhad celé trajektorie robota za současné minimalizace procesních a pozorovacích omezení. Algoritmy v této oblasti většinou vychází z metody *GraphSLAM* poprvé představené na přelomu tisíciletí. Druhou metodou je takzvané spojování menších map do *velké Submap matching*. Tato metoda je velmi dobrá v odstranění nepřesností spojených s kumulací chyby odometrie, a je proto dobře aplikovatelná ve velkém měřítku. Při spojování map předpokládáme často neznalost informací o poloze jednotlivých robotů. Způsoby spojování map se značně liší podle jejich typu, obecně se ale jedná o snahu hledání a identifikaci překryvů, které se poté využijí k fúzi map. [36, s. 7–8]

3.5.3 Vizuální SLAM

Využití obrazu pro SLAM může být považováno za velmi atraktivní především díky širokému rozsahu získatelných informací jako vzhled prostředí, barva, textura pro úkoly detekce a klasifikace. V porovnání s jinými technologiemi pro podobné účely jsou kamery levnější, lehčí a mají menší spotřebu energie. Na druhou stranu je potřeba adresovat možné problémy s nedostatečným rozlišením, změnou světelných podmínek, povrchy s nedostatkem struktury nebo rozmazáním snímků za pohybu. Je tedy potřeba kromě příslušných změn v konfiguraci provést i kalibraci na dané prostředí a podmínky. Další možné problémy jsou spojeny s množstvím kumulovaných chyb nebo nekonzistentní tvorbou mapy, což může mít různé příčiny. První možnou příčinou je uvažování hladkého pohybu kamery v průběhu snímání, jakékoliv náhlé změny rychlosti pohybu nebo výchylky s sebou zákonitě přinesou řadu nepřesností ve výsledném zpracování. Druhým předpokladem společným pro většinu zde uváděných algoritmů je neměnnost prostředí, v případě využití v prostředí dynamickém je třeba tento faktor náležitě zohlednit. Třetím problémem, se kterým se můžeme setkat hlavně v aplikacích na velkém území, je složení světa z mnoha opakujících se elementů (semafory, kamenné bloky, zdi, …), což značné komplikuje rozpoznání dříve zmapovaných oblastí.
[47, s. 2–5]

Obecný koncept současné lokalizace a mapování z obrazu se skládá ze tří částí. V první části inicializace jsou zavedeny globální souřadnice potřebné k dalším dvou krokům – sledování a mapování. Sledování má za úkol určovat pozici senzoru pomocí tvorby 2D nebo 3D korespondencí mezi obrazem a mapou. Tento úkol je v literatuře nazýván jako PnP problém (perspective-n-points) a je řešen například *RANSAC* nebo *EPnP* algoritmem. Mapování se zabývá výpočtem a rozšiřováním 3D struktury při pohybu kamery a jeho výsledkem je 3D model s množstvím detailů odpovídajícímu použité technice. Obdobou odometrie, která využívá enkodérů na kolech pro odhad a určení pozice robota v prostoru, je takzvaná vizuální odometrie se stejným cílem, akorát s využitím pouze analýzy obrazu. *vSLAM* na rozdíl od vizuální odometrie uvažuje pozici v globálním hledisku a výsledky jsou v něm doplněny například uzavíráním smyčky. [45, s. 2–3] [46, s. 2–4]

Přístupy k *vSLAMu* se dají rozdělit do tří kategorií:

Pouze vizuální SLAM (Visual only SLAM) je založen pouze na zpracování 2D obrazu. Po získání obrazů z více úhlů pohledu, systém provede krok inicializace a definuje tím globální souřadnice. Ke svému fungování často používá pouze jen monokulární (jedna optická čočka) nebo stereo (dvě optické čočky vedle sebe) kameru. Monokulární kamera potřebuje vždy alespoň dva úhly pohledu k určení hloubky, oproti tomu stereo kamera dosahuje lepších výsledků, bohužel za cenu náročnějšího zpracování obrazů. Samotné hledání souvislostí mezi dvěma obrazy funguje na základě příznakových metod, kde dochází k detekci příznaků pro následné určení pozice nebo přímých metod, které využívají senzorická data bez předchozího zpracování a zabývají se souvislostmi mezi intenzitami pixelů a následnou minimalizací fotometrické chyby. [46, s. 4]

Vizuálně-inerciální SLAM (Visual-intertial SLAM) je přístup doplňující kamerová data o informace o pohybu a orientaci zařízení z IMU (Inertial Measurement Unit), která je tvořena kombinací gyroskopu, akcelerometru a magnetometru. Dva obvyklé postupy fúzování kamerových dat s IMU jsou volná nebo těsná fúze. Volná fúze nespojuje data z IMU pro odhad pozice, ale používá je pro odhad rotace a změn natočení kamery. Těsný přístup k fúzi využívá sloučení obou druhů dat do pohybové rovnice. [46, s. 5]

RGB-D SLAM využívá ke svému fungování RGB kameru s jednou čočkou doplněnou o senzor hloubky, není tedy třeba ji složitě dopočítávat. Většina těchto systémů využívá algoritmů typu ICP (iterativní nejbližší bod) pro lokaci senzoru a následnou fúzi pro rekonstrukci celé struktury. Snížení komplexnosti fáze inicializace je vykompenzováno potřebou velkého množství paměti a vysokou spotřebou energie. [46, s. 6]

4 Praktická část

Praktická část této diplomové práce se zaměřuje na aplikaci některých z výše zmíněných principů. Měl jsem možnost se podílet na rozsáhlejším projektu zabývajícím se mimo jiné robotizací a digitalizací ovocného sadu. Projekt je stále v rané fázi vývoje a z důvodů strategického charakteru některých informací a přístupů mi není dovoleno v tomto okamžiku sdílet všechny detaily. Hlavním záměrem této části je představení zadaných úkolů, problémů a řešení. Širší kontext bude poskytnut v případech, kde je to nutné pro adekvátní pochopení čtenářem.

4.1 Struktura

V současném stádiu je testovací ovocný sad monitorován pojízdným robotem vybaveným senzory. První úkol je tedy zaměřený na zpracování senzorických dat, tvorbu robotické mapy a extrakci klíčových informací o měření. Druhý a třetí úkol se zabývají návrhem přístupů pro poloautomatizaci a automatizaci měření. Nejprve bude podrobněji zkoumána využitá technologie z hlediska její hardwarové a softwarové stránky. Následující část je věnovaná řešení jednotlivých úkolů doplněných o nezbytné přílohy a vysvětlení. V závěrečné části budou shrnuty využité principy, zhodnocena jejich úspěšnost a naznačeny možné budoucí modifikace.

4.2 Využité technologie

Robotická platforma (obrázek č. 21) vyvinutá týmem České zemědělské univerzity je umístěna na podvozek Spider 3 Rider, který umožňuje individuální natáčení jednotlivých kol. Na vrchu platformy je usazená bedna s průmyslovým počítačem, RTK-GPS a inerciální měřící jednotka IMU. V přední části robota jsou vertikálně nad sebe umístěny jednotlivé kamery a 2D lidar (obrázek č. 22). Kola jsou vybavena enkodéry měřící otáčky jednotlivých kol pro tvorbu odometrické mapy.

Obrázek 21: Robotická platforma Spider 3 Rider

Obrázek 22: Kamerová instalace

4.2.1 Hardwarová část

Kamerový systém je tvořen několika kamerami, jejichž primárním cílem v této fázi je pouze snímat řadu stromů pro další analýzu a extrakci fotek jednotlivých stromů. Každá z těchto kamer má jiné rozlišení a jiné charakteristické vlastnosti

Obrázek 24: RouteCAM_P_S1G_CU20 [47]

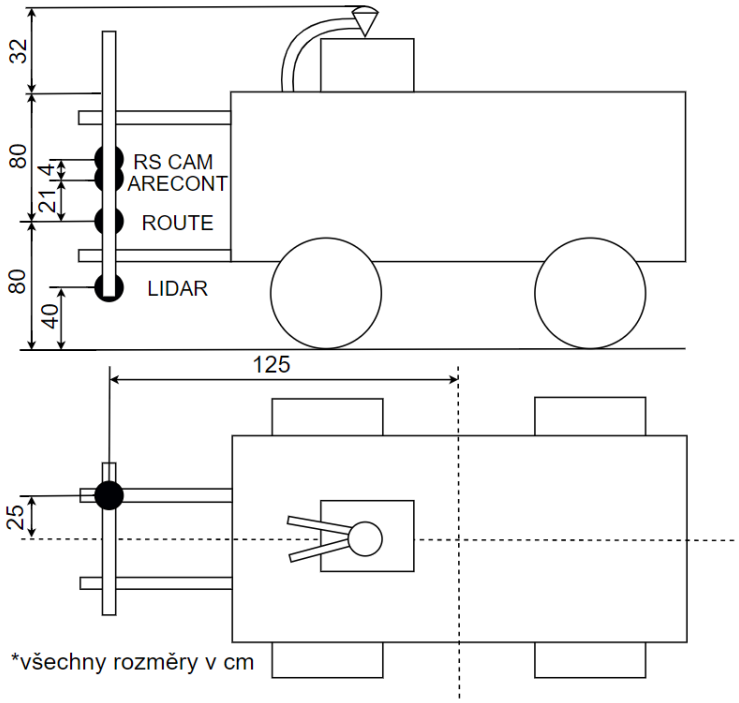
Obrázek 23: ArecontVision kamera AV3236DN [48]

Nejnovějším přírůstkem je kamera od firmy e-con Systems RouteCAM_P_S1G_CU20 (obrázek č. 23). Jedná se o 2 megapixelovou kameru připojenou přes RJ45 konektor umožňující napájení přes ethernet produkující snímky s rozlišením 1920 × 1080 pixelů s maximální frekvencí 30 snímků za sekundu. Je vybavena čočkou 1/2.7″ s ohniskovou vzdáleností 2,8 mm poskytující horizontální zorný úhel 102,69°. Druhou kamerou je 1,2 megapixelová AV3236DN od firmy ArecontVision napájená opět přes ethernetové rozhraní (obrázek č. 24). Poskytuje nižší rozlišení 1280 × 960 pixelů a díky čočce 1/3.2″ má zorný úhel 75,6°. Její hlavní výhodou je dobré fungování při nízkém osvětlení a v podmínkách vysokého kontrastu.

Obrázek 25: Intel RealSense hloubková kamera D455 [49]

Pro přidání další informace o vzdálenosti od objektu byla přidána i hloubková kamera od firmy Intel RealSense D455 (obrázek č. 25). Kamera v barevném režimu má rozlišení 1 megapixel a produkuje fotky s rozlišením 1280 × 800 pixelů v horizontálním zorném úhlu 90° s maximální frekvencí 30 snímků za sekundu. Optimální pracovní vzdálenost objektu od objektivu v hloubkovém režimu je 0,6–6 m. Hloubkový režim poskytuje rozlišení 1280 × 720 pixelů s frekvencí až 90 snímků za sekundu. Komunikace a napájení jsou řešené pomocí USB-C 3.1. Roli dálkoměru v tomto zapojení zastává 2-D lidar TIM571 od společnosti Sick (obrázek č. 26). Jeho výstupem jsou dálková měření v zorném poli 270° s úhlovým rozlišením 0,33°. Měřené vzdálenosti v pracovním pásmu 0,05–25 m jsou posílané přes ethernet s frekvencí 15 Hz. [50]

Obrázek 26: Sick Lidar TM571 [50]



Obrázek 27: Rozmístění senzorů na robotovi

Obrázek č. 27 zobrazuje umístění jednotlivých senzorů v rámci celého robota. Lidar je umístěn v přední části ve výšce 40 cm s cílem detekovat kmeny stromů a větší překážky v cestě. Jednotlivé kamery jsou uloženy vertikálně nad sebou na tyči na pozicích respektujících jednotlivé vertikální zorné úhly pro zabrání celých stromů.

4.2.2 Softwarová část

Pro sběr dat ze senzorů byl využit systém OSGAR (Open Source Garden Autonomous Robot) vyvinutý na České zemědělské univerzitě. Robot je v tomto prostředí reprezentován jako soubor jednotlivých propojených modulů řešících příslušné komponenty. Jednotlivá zařízení robota spolu komunikují po CAN sběrnici, která je připojená i na sériový port průmyslového palubního počítače. Hlavní výhodou tohoto přístupu kromě možnosti budoucího autonomního

provozu je systém logování všech dat pro analýzu v reálném čase nebo v off-line režimu. Každý posílaný datový blok obsahuje časové razítko, identifikátor datového toku, velikost posílaných dat a neupravená data. [51]

4.3 Datová fúze senzorických dat

Cílem fúze senzorický dat (kamerové snímky, IMU, GPS, lidar, odometrie) v tomto případě je primárně extrakce informací o pozici stromů a uložení snímků stromů do databáze pro archivaci a další zpracování. Detailní uložení dat o vozidle a senzorech prostřednictvím logů umožňuje tvorbu algoritmů a dalšího zpracování v off-line módu bez ztráty klíčových informací.

4.3.1 Struktura programu

Ke zpracování programovací části byl využit jazyk Python, v němž je napsána klíčová knihovna OSGAR. V této části bude popsána zvolená struktura programu a jednotlivý tok dat mezi jednotlivými bloky, kde každý blok představuje jeden skript. Ve výsledném programu byla ponechána struktura zavedená při vývoji pro snadnější průběžnou vizualizaci a řešení nových problémů a požadavků při další práci na tomto projektu. Následující kapitoly budou popisovat jednotlivé bloky a tok jednotlivých datových proudů mezi nimi. Některé následující pasáže budou obsahovat i poznámky o možných modifikacích v případě aktualizace požadavků zadavatele práce. Kromě detailního popisu kódu přímo v programu budou myšlenky a záměry za zvolenou implementací popsány i v této části práce a bude tím vytvořená jakási dokumentace pro usnadnění budoucího používání a zavedení modifikací.

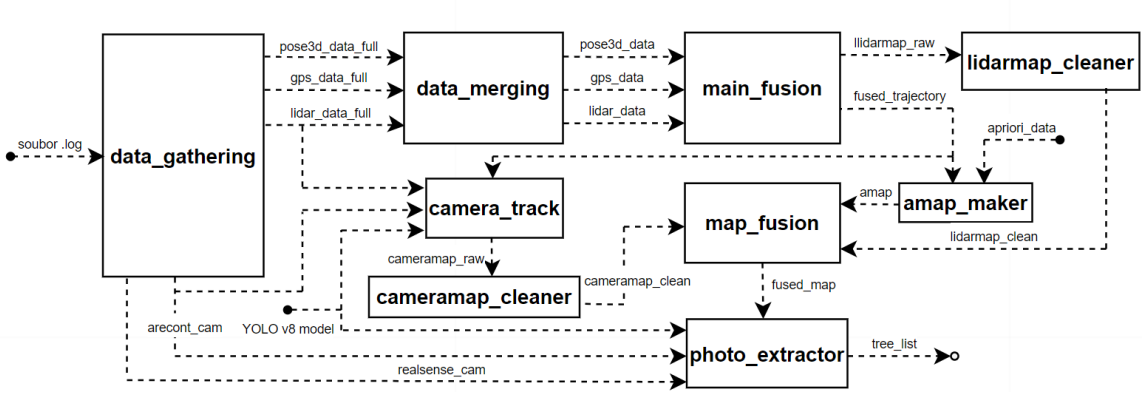
Prvním krokem po dokončení měření a jeho nahrání na server je analýza s využitím nástroje *osgar.logger*, který se stará o čtení i zapisováni do .log souboru. Analýza nám zobrazí názvy jednotlivých logovaných kanálů spolu s velikostí, počtem a frekvencí dat (obrázek č. 28). Využitá data jsou v dalším zpracování přejmenovaná pro větší přehlednost.

```
 k                          name      bytes | count | freq Hz
-------------------------------------------------------------
 0                           sys       3113 |    14 |   0.0Hz
 1             oak_camera.depth  795856236 |  1554 |   4.9Hz
 2             oak_camera.color  695403174 |  1553 |   4.9Hz
 3 oak_camera.orientation_list    1716268 |  1556 |   4.9Hz
 4                 arecont.image  195790614 |  1143 |   3.6Hz
 5              route_cam.image 1225856765 |  1905 |   6.0Hz
 6              qrdetec.qr_data          0 |     0 |   0.0Hz
 7           arduino_serial.raw          0 |     0 |   0.0Hz
 8           from_spider.pose2d      71157 |  6134 |  19.2Hz
 9        from_spider.rotation      100965 | 12394 |  38.7Hz
10           from_spider.pose3d     404844 |  6134 |  19.2Hz
11             from_spider.scan    5578911 |  2783 |   8.7Hz
12        from_spider.position      68390 |  6218 |  19.4Hz

Total time 0:05:19.943740
```

Obrázek 28: Analýza záznamu pomocí osgar.logger

Zmiňovaná struktura programu a tok dat mezi jednotlivými bloky jsou naznačeny ve schématu (obrázek č. 29). Data z logu a apriorní znalosti jsou použity k tvorbě tří na sobě nezávislých map, které jsou spojeny dohromady pro vytvoření finální mapy. Ze znalostí o poloze jednotlivých stromů v prostoru jsou pak získány jejich ořezané fotky a jsou uloženy do databáze ve formě listu obsahujícího všechny požadované charakteristiky.



Obrázek 29: Návrhové schéma programu

4.3.2 Sběr a úprava dat

Hlavním vstupem do celého programu je soubor .log s uloženým záznamem o měření. Skript *data_gathering* je zaměřen na sběr dat v požadovaném formátu a ukládání do složky pro další využití. Tento přístup byl zvolen, jelikož sběr dat může u rozsáhlého měřeni při načítání hlavně kamerových snímků zabrat dlouhou dobu. O ukládání dat a jejich načítání pro další využití se stará python knihovna *joblib*. Čtení dat z logu ve smyčce je zprostředkováno třídou z OSGAR balíčku *LogReader.* Smyčka prochází jednotlivé řádky záznamu, hledá příslušné identifikátory datových toků a získá data doplněná o časová razítka. Získaná data jsou rozšifrována, převedena do vhodného formátu a postupně ukládána do datových struktur. Při měření probíhá ukládání do záznamů dat z jednoho senzoru vždy nezávisle na ostatních s vlastní frekvencí. Tabulka 3 zobrazuje jednotlivé listy po jejich formátové i sémantické stránce.

Tabulka 3: Data získaná pro další zpracování

Název listu	**Formát prvku**	**Vysvětlení**
Gps_data	[čas, [E, N]]	GPS souřadnice v stupních ve formátu [délka, šířka]
Pose3d_data	[čas, [x, y, z, yaw]]	x, y, z souřadnice získané z odometrie v milimetrech doplněné o úhel natočení “yaw”
Lidar_data	[čas, [l1...l811]	Informace o délce v rozpětí 0–270 ° s rozlišením 0,33˚ v milimetrech
Rscolor_data	[čas, numpy pole]	Každý pixel v poli je popsán pomocí trojice hodnot reprezentující intenzitu barevných kanálů (RGB). Barevná RealSense kamera.
Arecontcam_data	[čas, numpy pole]	Barevná ArecontVision kamera

Pro následné zpracování a využití informací dohromady je třeba data v čase ucelit, o což se stará skript *data_merging.* Vstupem jsou *gps_data, pose3d_data* a *lidar_data.* Jeden z přístupů je spojit vždy nejbližší hodnoty jednotlivých měření, tím se ale do systému vnesou zbytečné odchylky už na začátku programu. Vzhledem k známé trajektorii robota – snaha o rovnoměrný přímočarý pohyb si můžeme dovolit chybějící hodnoty pozice robota odhadnout. Jako časové reference nám zde budou sloužit lidarová měření, k nimž dopočteme pose3d a GPS informace. Pro každé časové razítko lidarových měření nalezneme nejbližší vyšší a nejbližší nižší razítko u informací o pozici a proložíme tyto body přímkou. Nalezený bod na přímce odpovídajícímu referenčnímu času je poté uložen jako nová hodnota. Tímto způsobem se postupně dopočtou všechny hodnoty. Obecná lineární interpolace je popsána následujícím vztahem:

$$f(x) = f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0), pro\ x_0 < x < x_1 \quad (14)$$

Využití přístupu lineární informace dává smysl v případech, kdy nedochází k velkým skokovým změnám a kdy předpokládáme lineární vývoj, což je v tomto případě aplikovatelné pouze na informace o pozici. Výsledná data se sjednocenými časovými informacemi jsou uložena pomocí knihovny joblib do příslušných datových struktur pro další používání.

4.3.3 Main_fusion a lidarmap_cleaner

Časově sjednocená data vstupují do bloku *main_fusion*, kde probíhají současně dva hlavní procesy:

- fúze informací o pozici robota,
- tvorba mapy na základě lidarových dat.

Třetí část této kapitoly se bude zabývat blokem *lidarmap_cleaner*.

Fúze informací o pozici robota

Informace o pozici robota je dána GPS senzorem a odometrickými informacemi. Enkodéry měří natočení jednotlivých kol a na základě rozdílu natočení a vzdálenosti kol se v OSGARu urči úhel a ujetá vzdálenost, která je převáděna díky známému úhlu na přírůstky v ose x a v ose y. Postupným přičítáním těchto přírůstků vznikne výsledná trajektorie. Přírůstek nebo úbytek v ose z je počítán z absolutního natočení auta (když gyroskop vyhodnotí, že robot jede z kopce, hodnota souřadnice z klesá). Nevýhodou tohoto přístupu je kumulace odchylek, které značně rostou hlavně v případě prokluzů kol. Použití odometrie při dlouhých vzdálenostech s sebou nese značné množství nakumulovaných odchylek a není vhodné jí úplně důvěřovat. Na kratších vzdálenostech ale své využití má a v naší aplikaci bude slučována spolu s pozicí získanou GPS.

V současné verzi řízení robota při startu měření robot resetuje svou pozici a natočení a začíná od nulových hodnot. Vzniká zde tedy problém se souřadnicovými systémy. První souřadnicový systém je definován GPS daty, kde osa x odpovídá zeměpisné délce (v našem případě východní) a osa y zeměpisné šířce (v našem případě severní). Druhý souřadnicový systém je definovaný v momentě, kdy se spustí měřicí program a třetí souřadnicový systém je v každém bodě robota respektující jeho natočení. Nedostatkem měřených informací je absence informací v záznamu o počátečním natočení z kompasu vůči severu. Při přehrání lidarových dat pomocí nástroje *lidarview* v OSGARu, jsou vidět dvě řady stromů mezi kterými robot projíždí a relativní pozice robota vůči nim. Jelikož jsou řady statické, lze určit přibližná trajektorie, po které robot jel. Tato trajektorie, co se výchylek robota vůči přímočarému pohybu týče, odpovídá mnohem více průběhu zachycenému senzorem GPS. Zároveň jsou pomocí GPS měřeny i jednotlivé souřadnice sloupů umístěných ve stromové řadě. Jako reference pro spojení informací z GPS a odometrie budou tedy sloužit GPS data.

```
def rotation_angle(point_1, point_2):
    #Vypocet vektoru
    vector_1 = (point_1[0], point_1[1])
    vector_2 = (point_2[0], point_2[1])
    #Skalarni soucin vektoru
    skal_souc = vector_1[0]*vector_2[0] + vector_1[1]*vector_2[1]
    #Velikost vektoru
    vector_1_lenght = math.sqrt(vector_1[0]**2 + vector_1[1]**2)
    vector_2_lenght = math.sqrt(vector_2[0]**2 + vector_2[1]**2)
    angle_rad = math.acos(skal_souc/(vector_1_lenght*vector_2_lenght))
    angle_deg = math.degrees(angle_rad)
    #Rozhodnuti o smeru rotace
    if vector_1[0]*vector_2[1] - vector_1[1]*vector_2[0] < 0:
        angle_deg = 360 - angle_deg
    return -angle_deg

def rotate(x, y, angle_deg): #Kladny smer po smeru hodinovych rucicek
    x_new = x*math.cos(math.radians(angle_deg))-
            y*math.sin(math.radians(angle_deg))
    y_new = x*math.sin(math.radians(angle_deg))+
            y*math.cos(math.radians(angle_deg))
    return [x_new, y_new]
```

Zdrojový kód 1: Funkce rotation_angle() a rotate()

Pro zjištění úhlu rotace byla vytořena funkce *rotation_angle()*, kde jsou vstupem dva body. Funkce poté oba body propojí s počátkem souřadnicového systému a určí svíraný úhel (zdrojový kód č. 1). Vypočtený úhel je použitý jako jeden ze vstupů do funkce *rotate()*. Bod vložený spolu s úhlem jako vstup je pak rotován o příslušný úhel a jeho souřadnice jsou patřičně přepočítány.

Souřadnicový systém, který je zvolen jako hlavní pro celý zbytek práce má stejné natočení jako GPS, kde kladná část osy y směřuje k severu a kladná část osy x směřuje k východu. Počátek tohoto souřadnicového systému byl zvolen v počátku měření. GPS a odometrické informace jsou zde vynášeny jako rozdíly v jednotlivých osách oproti počátku.

```
#Zacatecni a koncovy bod
start_gps = [gps_data[0][1][0], gps_data[0][1][1]]
start_pose3d = [pose3d_data[0][1][0], pose3d_data[0][1][1],pose3d_data[0][2]]
end_gps = [gps_data[-1][1][0], gps_data[-1][1][1]]
end_pose3d = [pose3d_data[-1][1][0], pose3d_data[-1][1][1],
              pose3d_data[-1][2]]
#Rozdil mezi zacatkem a koncem v jednotlivych osach
diff_end_gps = [distance([start_gps[1], start_gps[0]], [start_gps[1],
                gps_data[-1][1][0]]).m * 1000,
                distance([start_gps[1], start_gps[0]], [gps_data[-1][1][1],
                start_gps[0]]).m * 1000]
diff_end_pose3d = [pose3d_data[-1][1][0] - start_pose3d[0],
                   pose3d_data[-1][1][1] - start_pose3d[1],
                   pose3d_data[-1][2] - start_pose3d[2]]
#Vypocet uhlu rotace
ang_rot = rotation_angle([diff_end_gps[0], diff_end_gps[1]],
          [diff_end_pose3d[0], diff_end_pose3d[1]])
```

Zdrojový kód 2: Získání úhlu pro rotaci

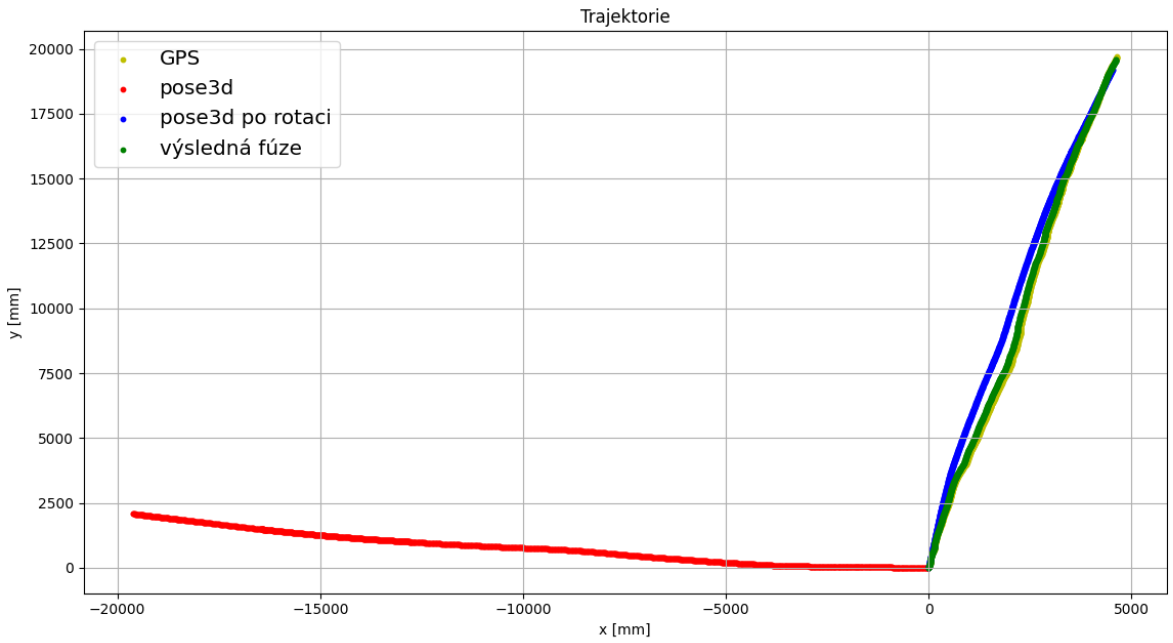
Před smyčkou přes všechna data jsou načteny počáteční pozice *start_gps* a *start_pose3d* a koncové pozice měření *end_gps*, *end_pose3d* (zdrojový kód č. 2). Vzdálenost mezi začátkem a koncem měření se z odometrických dat získá prostým odečtením jednotlivých složek od sebe. Pro přepočet GPS souřadnic na milimetry je použita funkce *distance()* z knihovny *geopy*. Tímto sledem operací došlo k převedení obou trajektorií do stejného souřadnicového systému. To umožňuje na proměnné *diff_end_gps* a *dif_end_pose3d* popisující rozdíl v souřadnicích mezi startovním a koncovým bodem použít výše zmíněnou funkci *rotate()* pro získání úhlu natočení.

```
#Rozdily oproti predchozim hodnotam
diff_prev_gps = [distance([gps_data[a-1][1][1], gps_data[a-1][1][0]],
                          [gps_data[a-1][1][1], gps_data[a][1][0]]).m * 1000,
                 distance([gps_data[a-1][1][1], gps_data[a-1][1][0]],
                          [gps_data[a][1][1], gps_data[a-1][1][0]]).m * 1000]
diff_prev_pose3d = [pose3d_data[a][1][0] - pose3d_data[a-1][1][0],
                    pose3d_data[a][1][1] - pose3d_data[a-1][1][1],
                    pose3d_data[a][2] - pose3d_data[a-1][2]]
diff_prev_pose3d_R = rotate(diff_prev_pose3d[0], diff_prev_pose3d[1], ang_rot)
#Spojeni odometrickych a gps informaci na zaklade vazeneho prumeru
fused_prev_move=diff_prev_pose3d_R[0]*weight_pose3d+diff_prev_gps[0]*weight_gps,
                diff_prev_pose3d_R[1]*weight_pose3d+diff_prev_gps[1]*weight_gps]

fused_position =[fused_position[0]+fused_prev_move[0],
                 fused_position[1]+fused_prev_move[1]]
#Pripnuti finalni pozice do listu
fused_trajectory.append(fused_position)
```

Zdrojový kód 3: Fúze pozice robota ve smyčce

Hlavní částí tohoto skriptu je smyčka přes všechna lidar měření společná pro fúzi pozice i tvorbu mapy z lidarových dat. Pro každé měření jsou spočítané rozdíly posunu v obou osách oproti předchozím hodnotám a informace získané z odometrie jsou rotovány o úhel *ang_rot()* *(*zdrojový kód č. 3). Následně dojde ke spojení informací s využitím váženého průměru. V průběhu smyčky jsou jednotlivé přírůstky přičítány do výsledné proměnné a ukládány do listu.

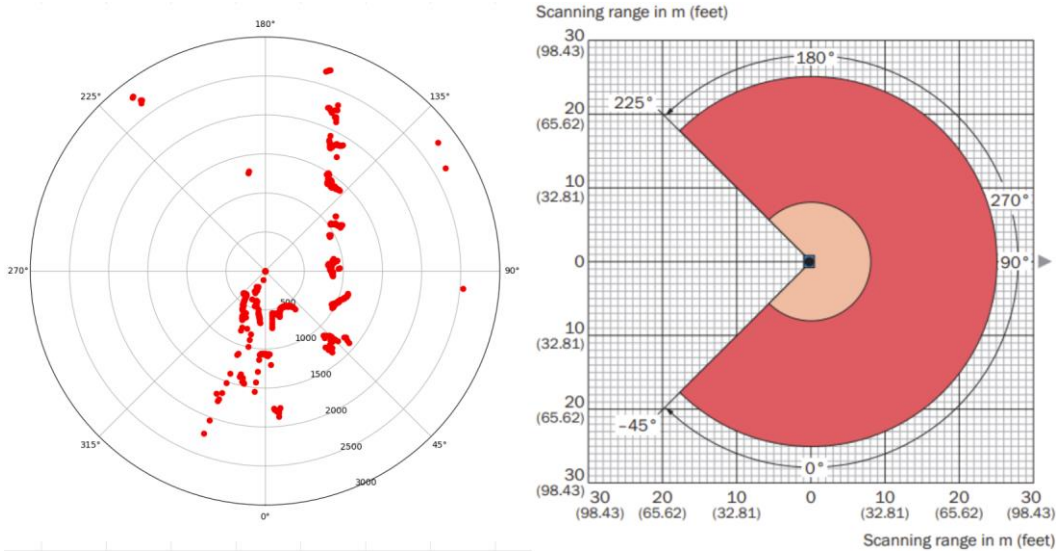


Graf 1: Trajektorie robota

Graf č. 1 názorně zobrazuje jednotlivé zmiňované trajektorie. Červeně je zde vynesená pozice získaná z odometrie před rotací a modře po rotaci. Žlutá barva reprezentuje informace z GPS senzoru a zelená výsledný průběh získaný váženou fúzí. Je zde vidět, že průběžná pozice robota uložena v kanálu pose3d adekvátně nezaznamenává výchylky z přímočarého pohybu a výsledná trajektorie má tak rovnější charakter, než jak tomu bylo ve skutečnosti. I přes to mají informace o ujeté vzdálenosti pro naší aplikaci význam. Cílem je v ideálním případě vytvoření robustního systému, který neselže, i když v nějakých částech informace z jednoho zdroje nebudou dostupné. Důvěra experta ve správnost naměřených dat je reprezentována pomocí vah jednotlivých kanálů. Pro toto měření byly zvolena váha 0,8 pro informace poskytnuté GPS a 0,2 pro odometrii. Všechny další procesy vyžadující pozici robota v čase budou v této aplikaci používat výslednou fúzovanou trajektorii.

Tvorba mapy z lidarových dat

Druhou částí bloku *main_fusion* je analýza lidarových dat v čase a tvorba mapy sadu. Lidar měření v každém bodě obsahuje soubor vzdáleností, které odpovídají jednotlivým úhlům v rozpětí 0–270° s úhlovým rozlišením 0.33°.



Obrázek 30: Lidar snímek v polárních souřadnicích v pracovní oblasti [51]

Pro přehlednost vizualizace nezpracovaných dat lidaru v polárních souřadnicích byl graf na obrázku č. 30 omezen na vzdálenost 3 metry. Pravá část obrázku představuje pracovní diagram přímo z katalogu výrobce. Pohyb robota je ve směru odpovídajícímu 180° a řada shluků v pravé části reprezentuje jednotlivé kmeny stromů. Pro nasazení algoritmu pro identifikaci shluků a jejich center je potřeba data předzpracovat. Prvním krokem je omezení se pouze na specifickou výseč pro specifickou vzdálenost. Vybrané body jsou následovně převedeny z polárních souřadnic na kartézské. O předzpracování se ve smyčce stará část kódu naznačená ve zdrojovém kódu č. 4, kde parametry *angle_range_low* a *angle_range_high* zajišťují výběr zvolené výseče a parametry *max_range* a *min_range* omezují vzdálenost pro správné nalezení shluků.

```
A = lidar_data[a][1] #Soubor lidarovych mereni
#Prevod polarnich dat na xy souradnice
#omezeni parametry: angle_range_low, angle_range_high - uhel
#                 : min_range, max_range - vzdalenost
for l in range(angle_range_low, angle_range_high, 1):
    if min_range < A[1] < max_range:
        #Rozdeleni na jednotlive kvadranty
        if (-45 <= -45 + dif_deg * l) & (-45 + dif_deg * l < 0):
            merged_lidar_short_xy.append(
            [-A[1] * math.sin(math.radians(45 - dif_deg * l)),
             -A[1] * math.cos(math.radians(45 - dif_deg * l))])
        elif (0 < -45 + dif_deg * l) & (-45 + dif_deg * l < 90):
            merged_lidar_short_xy.append(
            [A[1] * math.sin(math.radians(-45 + dif_deg * l)),
            -A[1] * math.cos(math.radians(-45 + dif_deg * l))])
        elif (90 < -45 + dif_deg * l) & (-45 + dif_deg * l < 180):
            merged_lidar_short_xy.append(
            [A[1] * math.cos(math.radians(-135 + dif_deg * l)),
             A[1] * math.sin(math.radians(-135 + dif_deg * l))])
        elif (180 < -45 + dif_deg * l) & (-45 + dif_deg * l <= 225):
            merged_lidar_short_xy.append(
            [-A[1] * math.sin(math.radians(-225 + dif_deg * l)),
             +A[1] * math.cos(math.radians(-225 + dif_deg * l))])
        else:
            print("Out of lidar FOV")
    else:
        continue
```

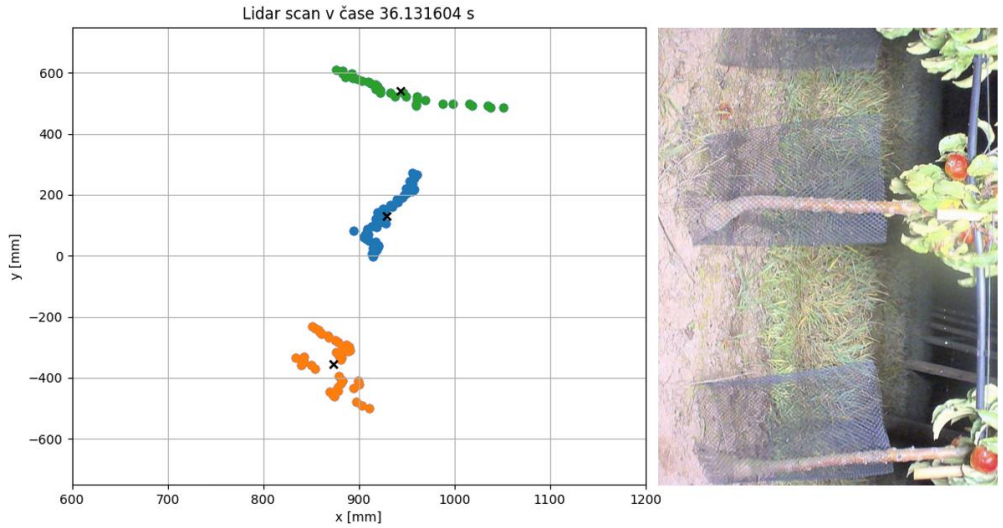
Zdrojový kód 4: Předzpracování lidarových dat

Pokud bod v setu měření projde nastavenými podmínkami, je v závislosti na kvadrantu, ve kterém se nachází, převeden do kartézských souřadnic a poslán k uložení do listu pro další zpracování. Na vytvořený list bodů je nasazen shlukovací algoritmus *mean-shift* nevyžadující apriorní počet shluků k určení (zdrojový kód č. 5). Pro praktickou aplikaci byla využita knihovna *sklearn* obsahující kromě samotného modelu i použité pomocné funkce. Konkrétní popis shlukovacích algoritmů byl poskytnut v rešeršní části, zkráceně se jedná o iterativní proces posouvání středu shluku k maximální hustotě bodů, dokud nedojde k dosažení optima nebo prahové hodnoty. Původní verze programu fungovala více na bázi robotického prozkoumávání, kde není počet shluků, co by měl algoritmus detekovat, známý. Pokud bychom chtěli použít apriorní mapu už ve fázi detekce stromů pro určení počtu stromů, které má algoritmus najít, dávalo by smysl využít spíše *k-means*. Výstupem shlukovacího algoritmu jsou *clusters*, obsahující rozdělení jednotlivých bodů poskytnutého pole a *cluster_centers* se souřadnicemi center. Podmínka ve spodní části obrázku kontroluje správnost detekce porovnání velikosti největšího nalezeného shluku s parametrem *maxclusterpoints*. Tento parametr je experimentálně určená hodnota omezující případ, kdy algoritmus nesprávně detekuje více shluků jako jeden celek.

```
# Mean-Shift clustering
X = np.array(merged_lidar_short_xy)  #Definice datasetu
bandwidth = estimate_bandwidth(X, quantile=0.25)
model = MeanShift(bandwidth=bandwidth)
yhat = model.fit_predict(X)
clusters = unique(yhat)
#Urceni souradnic stredu shluku
cluster_centers = model.cluster_centers_
#Nejvetsi velikost shluku
cluster_size = max(unique(yhat, return_counts=True)[1])
#Pokud je nejvetsi shluk moc velky, dalsi iterace
if cluster_size > maxclusterpoints:
    continue
```

Zdrojový kód 5: Mean-Shift algoritmus

Samotná fáze detekce stromů je znázorněna na obrázku č. 31, kde jsou jednotlivé identifikované shluky barevně odlišeny. Pravá strana obrázku odpovídá snímku z arecont kamery ve stejném čase. Můžeme zde vidět jednu z hlavních komplikací využití lidaru pro detekci kmenů v našem případě, a to je přítomnost pletiva chránícího mladé stromky proti škůdcům. Shluky lidarových bodů mají málokdy očekávaný kruhový charakter a kmínek stromu se může nacházet kdekoliv uprostřed shluku. Pokud by shluky stromů konzistentně měly část tvaru kruhu, mohli bychom na základě známých metod určit pozici stromu jako střed kruhu proloženého body [53]. Místo toho je souřadnice stromu určená jako souřadnice těžiště shluku odpovídající černým křížkům v grafu.



Obrázek 31: Identifikované stromy doplněné o kamerový snímek

Hlavní část kódu (zdrojový kód č. 6) se zabývá určením výsledných souřadnic centra v globálním souřadnicovém systému a datovou asociací. Každý určený shluk ze snímku (viz obrázek č. 31) je podrobený další analýze. Umístění lidaru nízko od země s sebou přináší riziko vysoké trávy nebo jiných překážek. Pokud není list bodů shluku dostatečně dlouhý, je s ním zacházeno jako s náhodnou překážkou nebo chybnou detekcí a je odfiltrován. Druhé omezení na daném snímku hledá vždy centra shluku s y-souřadnicí blízkou bodu 0, konkrétně na intervalu *<centerlowlim, centerhighlim>*. Toto omezení je zavedeno z následujících důvodů.

- Shluky s centry mimo definovaný interval jsou příliš vzdálené od kolmého pohledu na řadu stromů. Často tedy vidíme boční pohled na kmen obehnaný pletivem namísto čelního, což vnáší do systému nepřesnosti. Druhá a horší situace nastává, když je strom umístěný na pomezí limitu detekce lidarem. S pohybem auta v čase je takový strom definovaný klesajícím nebo rostoucím počtem bodů. To způsobuje, že vypočítaná centra jsou výrazně posunutá oproti skutečnosti.
- Díky známé vzdálenosti stromů z apriorních situací můžeme omezit interval *<centerlowlim, centerhighlim>* tak, aby se v něm nacházel vždy jen jeden strom, což značně usnadní identifikaci a asociaci dat mezi po sobě jdoucími měřeními.
- Identifikace stromů v bloku *camera_track* pomocí obrazového záznamu je omezena na střed obrazů kvůli problémům se zkreslením soudkovitostí. Tato problematika bude blíže vysvětlena v příští kapitole. Pro výslednou kombinaci vytvořených map do jedné výsledné dává smysl, aby individuální tvorba mapy probíhala na odpovídajících si datech. Z toho samého důvodu je snaha volit parametry *angle_range_low* a *angle_range_high* omezující oblast lidarového měření, tak aby výsledný úhel odpovídal zornému úhlu kamery používané k detekci.

```
# Iterace pres kazdy cluster
for index, cluster in enumerate(clusters, start=0):
    row_ix_list = where(yhat == cluster)[0].tolist()  # List bodu jednoho shluku
    # Minimalni velikost shluku k analyze
    if len(row_ix_list) > minclusterpoints:
        # Omezeni se na oblast ve stredu snimku
        if (cluster_centers[index][1] > centlowlim) &
           (cluster_centers[index][1] < centhighlim):
            center = cluster_centers[index]
                     #[lokal. centr. + umisteni senzoru    + pozice robota]
            real_center = [center[0] + sensor_placement[0] + fused_position[0],
                           center[1] + sensor_placement[1] + fused_position[1]]
            if len(centerlist_lidar) == 0:
                #Prvni detekce - pouze pripneme
                centerlist_lidar.append(
                [a, identifier, [real_center[0], real_center[1]]])
                #[ index a, identifikator, [centrum]]
            else:
                # Predikce pozice posledniho ulozeneho centra na novem snimku
                idd = centerlist_lidar[-1][1]
                ref_dist = [fused_trajectory[a][0] - fused_trajectory[idd][0],
                            fused_trajectory[a][1] - fused_trajectory[idd][1]]
                prev_center = centerlist_lidar[-1][2]
                pred_center = [prev_center[0], prev_center[1]]
                diff = math.dist(real_center, pred_center)
                # Porovnani vzdalenosti mezi predikci a vysetrovanym centrem
                if diff<ident_limit:
                    #Jedna se o posunute centrum, stejny identifikator
                    centerlist_lidar.append(
                    [a, identifier, [real_center[0], real_center[1]]])
                else:
                    #Jedna se o nove centrum, novy identifikator
                    identifier = identifier+1
                    centerlist_lidar.append(
                    [a, identifier, [real_center[0], real_center[1]]])
        else:
            continue
    else:
        continue
```

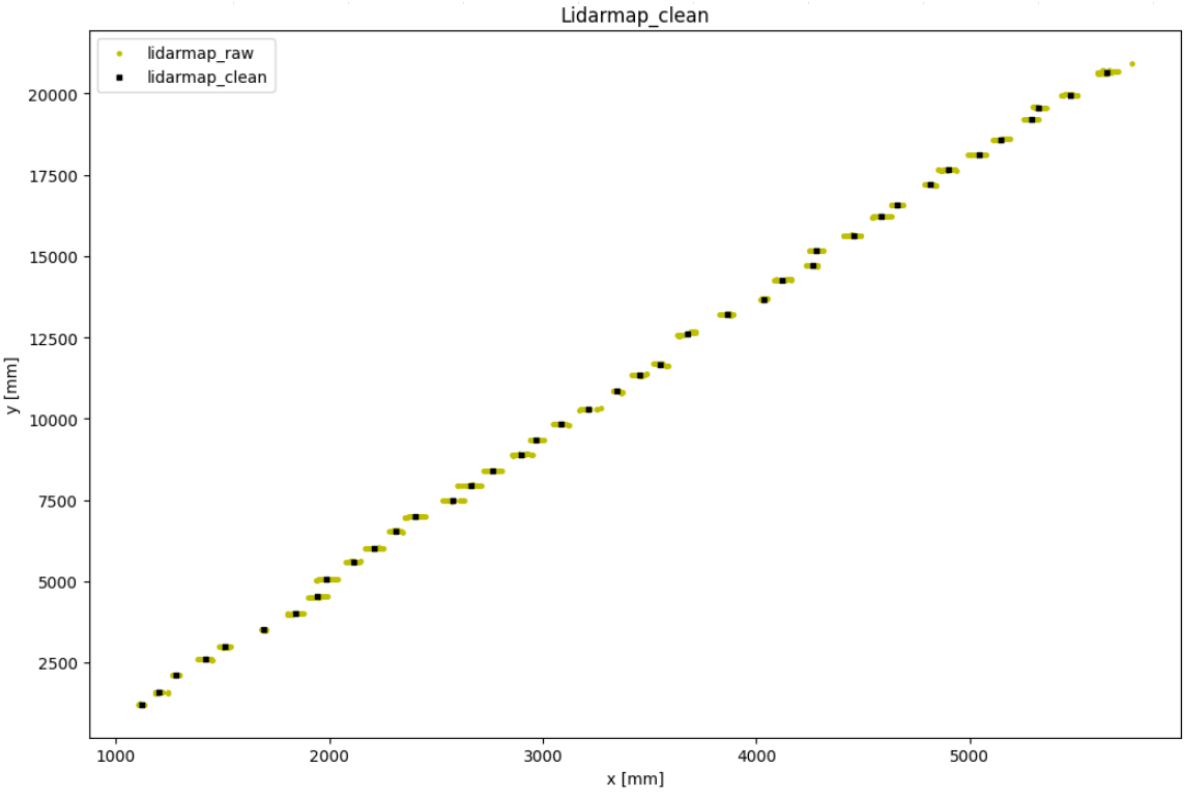
Zdrojový kód 6: Identifikace stromu a určení výsledné souřadnice

Pokud má detekované centrum minimální počet bodů a jeho lokální y-souřadnice se nachází v intervalu blízko středu, je shluk považován za správně detekovaný strom. Takový strom je převedený do globálního souřadnicového systému, spojením lokálních souřadnic detekovaného stromu, posunů v osách odpovídajícímu umístění senzorů na robotovi a globální pozici robota. Pokud se jedná o první detekci jakéhokoliv stromu, jsou souřadnice *real_center* spolu s identifikátorem stromu *identifier* a indexem odpovídajícímu měření pro pozdější určení času lidarového měření uloženy do listu *lidarmap_raw.* V případě, že se jedná o jinou, než první detekci rozhodujeme, zdali se jedná o centrum známé nebo nové. Pro tento účel je provedena predikce pozice posledního uloženého centra v současném snímku odečtením ujeté vzdálenosti ve směru jednotlivých os. Pokud se vzdálenost predikované pozice posledního uloženého centra od aktuálního vyšetřovaného centra odlišuje o více, než je hodnota parametru *ident_limit*, algoritmus vyhodnotí, že se jedná o nový strom, zvýší identifikátor a připne výslednou strukturu do listu. V případě, že je rozdíl predikce a reality menší, je struktura připnuta se stávajícím identifikátorem. Výstupy bloku *main_fusion* jsou listy *lidarmap_raw* obsahující listy detekovaných center a *fused_trajectory* s údaji o pozici robota v čase.

Lidarmap_cleaner

Tento blok je zaměřený na analýzu vstupního listu *lidarmap_raw* a tvorbu výsledné mapy. Přístup využitý při tvorbě mapy v main_fusion je získat co nejvíce měření jednotlivých stromů pro následné zprůměrování. První část tohoto skriptu extrahuje ze vstupního listu jednotlivé identifikátory a určuje jejich četnost v listu. Měření s identifikátory s malou četností jsou

odebrána a dále nejsou uvažována. Zbývající správné detekce podstupují proces, ve kterém se hodnoty pozice a času daného indexem *a* se stejným identifikátorem průměrují. Výstupem je list výsledných průměrů pozice a času pro četné identifikátory určují výsledné pozice stromů. Určení přibližného času měření, ve kterém je strom vidět na senzorech, je důležité hlavně pro část extrakce fotek.



Graf 2: Mapa z lidarových dat

Ze vstupu *lidarmap_raw* vyneseného žlutou barvou byla vyfiltrována jednotlivá centra označená černým čtvercem (graf č. 2). Problém tohoto přístupu je obtížnost rozeznání mezi sloupem vodící konstrukce a kmenem stromu. Výsledek je bohužel úzce spojený s nastavenými parametry vyžadující apriorní znalosti. Místa ve výsledné mapě, kde bychom strom očekávali, ale algoritmus tam žádný nedetekoval, jsou s nejvyšší pravděpodobností případy, kdy shlukovací algoritmus nesprávně určil shluky a tato data nebyla díky podmínce zahrnuta.

Následující text se bude zaobírat jednotlivými parametry, jejich dopadem na výsledek algoritmu a jejich optimálními hodnotami nastavení.

Tabulka 4: Parametry identifikace lidarem

Název parametru	**Stručný popis**	**Komentář k nastavené hodnotě**
Min_range, Max_range	Minimální a maximální detekovaná vzdálenost lidarem	Výchozí: *min_range* = 400 mm, *max_range* = 1700 mm Zvolena oblast, ve které jsou vidět tři až čtyři stromy kvůli nejlepším výsledkům shlukování
Angle_range_low, angle_range_high	Spodní a horní hranice intervalu úhlu lidaru	Výchozí: *angle_range_low* = 315, *angle_range_high* = 540 Hodnota odpovídá trojnásobku úhlu, analyzovaná výseč má úhel 75°.

Centerlowlim, Centerhighlim	Spodní a horní odchylka od osy y	Výchozí: *centerlowlim* = -150 mm, *centerhighlim* = 150 mm Omezení se na střed snímku pro sledování stromů v průběhu času. Další zmenšení intervalu znamená riziko menšího počtu detekcí pro výsledné průměrování.
Minclusterpoints, maxclusterpoints	Minimální a maximální počet bodů ve shluku	Výchozí: *minclusterpoints* = 10, *maxclusterpoints* = 120
Ident_limit	Limitní vzdálenost pro určení identifikátoru stromu	Výchozí: *ident_limit* = 150 mm

4.3.4 Camera_track a cameramap_cleaner

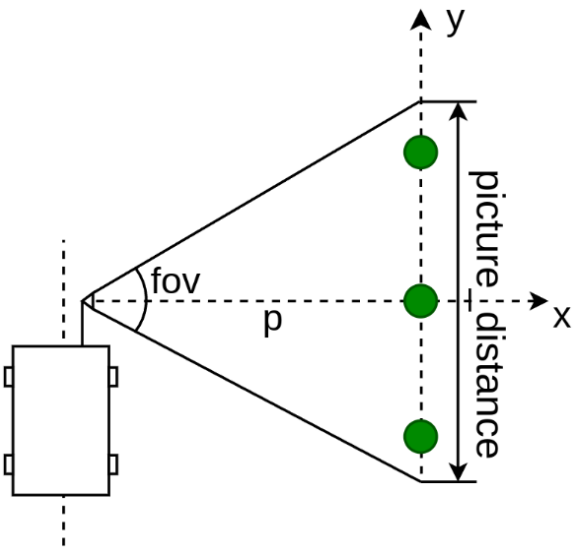
Tato kapitola se zabývá podobným úkolem jako předchozí kapitola, což je tvorba mapy, tentokrát ale z analýzy kamerových snímků.

4.3.4.1 Camera_track

Hlavním způsobem detekce stromů je v tomto případě algoritmus YOLO v8 založený na konvoluční neuronové síti natrénovaný jiným účastníkem projektu. Vstupem do bloku *camera_track* je list obsahující informace o trajektorii robota *fused_trajectory*, list obsahující lidarová data pro určení vzdálenosti od řady stromů *lidar_data_full*, snímky z arecont kamery *arecont_cam* a natrénovaný detektor YOLO v8.

První část programu využívá principů z bloku *data_merging*, pro sjednocení časových informací mezi lidarem a kamerovými daty. Pro jednotlivá časová razítka kamerových snímků jsou pomocí funkce *find_nearest()* nalezeny nejbližší časová razítka spojené pozice a její hodnoty. Na základě rovnice č. 14 jsou poté chybějící hodnoty dopočítány interpolací. V této fázi máme každý kamerový snímek doplněno údajem o pozici robota.

Hlavní algoritmus operuje na bázi smyčky procházející všechny snímky z kamery, na jejíž snímky je neuronová síť primárně natrénována pro detekci kmenů stromů. Po detekci stromu jsou jeho souřadnice odečteny z pozici na obraze a z měření vzdálenosti stromové řady určené lidarem. Závislost mezi využívanými parametry a konstantami je vyjádřena na obrázku č. 32.



Obrázek 32: Schéma měření

Rovnice č. 15 slouží k výpočtu zorného úhlu fov (Field of View) ze známých hodnot ohniskové vzdálenosti čočky *f* a šířky senzoru *S*.

$$fov = arctan\left(\frac{S}{2 \cdot f}\right) \tag{15}$$

P značí vzdálenost robota od stromové řady a je vypočtena jako průměrná hodnota x-ových vzdáleností nalezených shluků. Vzdálenost *picture_distance* je dopočtena podle vztahu č. 16

$$picture\_distance = 2 \cdot p \cdot tan\left(\frac{fov}{2}\right) \tag{16}$$

Hodnota *picture_distance* je poté použita jako vstup do funkce *generate_y()*, která přiřadí souřdnici y detekovanému kmenu. V tomto vztahu č. 17 reprezentuje x pozici stromu na snímku a šířka snímku je uváděna v pixelech.

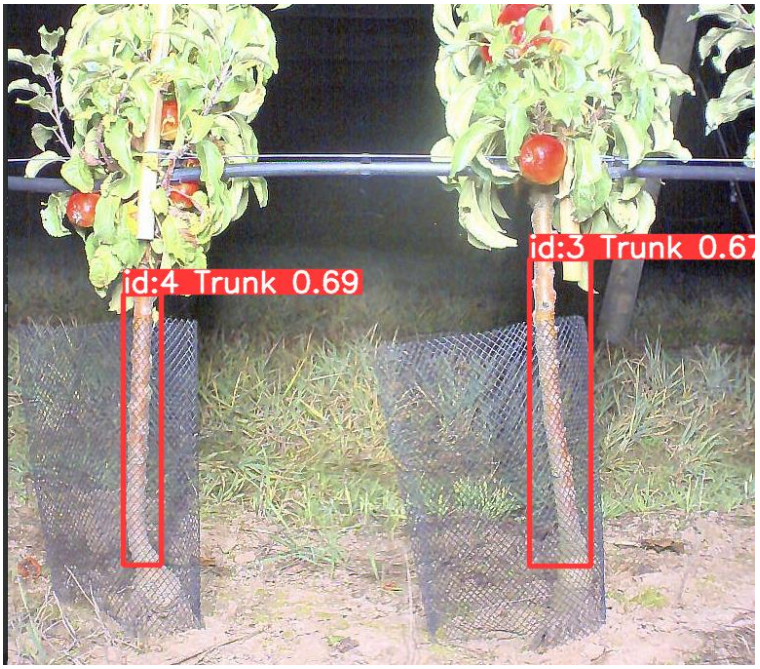
$$y = -\frac{x}{\check{s}\acute{i}\check{r}ka\ sn\acute{i}mku} \cdot picture\_distance + \frac{picture\_distance}{2} \tag{17}$$

```
frame = arecontcam_data[a][1] #Nacteny snimek
frame = cv2.rotate(frame, cv2.ROTATE_90_CLOCKWISE) #Rotace snimku
#Vysledek detekce a sledovani
results = model.track(frame, persist=True, conf=0.25, iou=0.5)
#Vizualizace vysledku
annotated_frame = results[0].plot()
#Pixelove souradnice detekovaneho boxu
boxes = results[0].boxes.xyxy
for m in range(len(boxes)):
    #X souradnice stredu boxu
    xbox = float(boxes[m][0]+boxes[m][2])/2
    #Parametr duvery v detekci
    confbox = results[0].boxes.conf[m].item()
    #Identifikator stromu
    boxid = results[0].boxes.id
    #Omezeni se na stred stnimku
    if xbox > distortion_limit and xbox < arecont_res[1]-distortion_limit:
        if boxid != None:
            boxid = int(results[0].boxes.id[m].item())
            #Souradnice stromu
            tree_coord = [p + sensor_placement[0] + position_arecontcam[a][0],
                          generate_y(xbox, picture_distance) + sensor_placement[1]
                          + position_arecontcam[a][1]]
            cur = [confbox, boxid, [tree_coord[0], tree_coord[1]],
                   arecontcam_data[a][0].total_seconds(), a]
            all_trees.append(cur)
        else:
            #Situace, kdy strom nema prideleny identifikator
            #Popsano v dalsi casti prace
    else:
        print("Mimo rozsah ohraniceni")
```

Zdrojový kód 7: Detekce kmenu a přiřazení souřadnic

Pro každý snímek kamery probíhá detekce a sledování jednotlivých stromů. Zdrojový kód č. 7 popisuje tento proces po stránce programu. Výsledek detekce kmenu stromu ve formě třídy z načteného snímku je uložen do proměnné *results*. Vizualizace výsledku analýzy jednoho obrázku je zobrazena na obrázku č. 33. Následující část kódu prochází jednotlivé detekované boxy vyznačené červenou barvou. Snímky z kamery jsou hlavně v krajních částech značně zkreslené soudkovitostí, proto je zaveden interval ve středu snímku pomocí parametru *distortion_limit*. Pokud se hodnota pixelu odpovídající střední hodnotě horizontální pozice boxu nachází tomto intervalu jsou stromu přiřazeny souřadnice v globálním souřadnicovém systému. X-ová složka je definovaná průměrnou vzdáleností stromové řady a y-ová souřadnice je výstupem funkce *generate_y()*. K těmto hodnotám je přičtena pozice senzorů vůči centru robota a pozice samotného robota v daném čase. Výsledné souřadnice stromu *tree_coord,*

jsou doplněny hodnotami o důvěře ve správnost detekce *confbox*, identifikátoru boxu *boxid*, času, ve kterém byla fotografie pořízena, a index a odkazující na pozici v listu a.



Obrázek 33: Vizualizace detekce kmenu

Soudkovitost se dá ze snímku odstranit pomocí kalibrace kamery a následných transformací. Jednotlivé potřebné parametry jako je matice vnitřních parametrů kamery a vektor koeficientů zkreslení jsou uložené v přiložených „.json souborech“. O samotné odstranění soudkovitosti se stará funkce *undistort()* z knihovny *opencv*. Soudkovitost je tímto způsobem odstraněna za cenu ztráty rozlišení. Problémem tohoto přístupu je oříznutí značné části kmenů používaných k detekci stromů. Z toho důvodu není část kódu, která se o korekci soudkovitosti stará využita a je v programu ponechána pro budoucí implementaci. Místo toho je využit pouze středový interval snímku. Výstupem je *cameramap_raw* obsahující souřadnice stromů, identifikátory, časové hodnoty a pravděpodobnostní koeficienty detekce.

4.3.4.2 Cameramap_cleaner

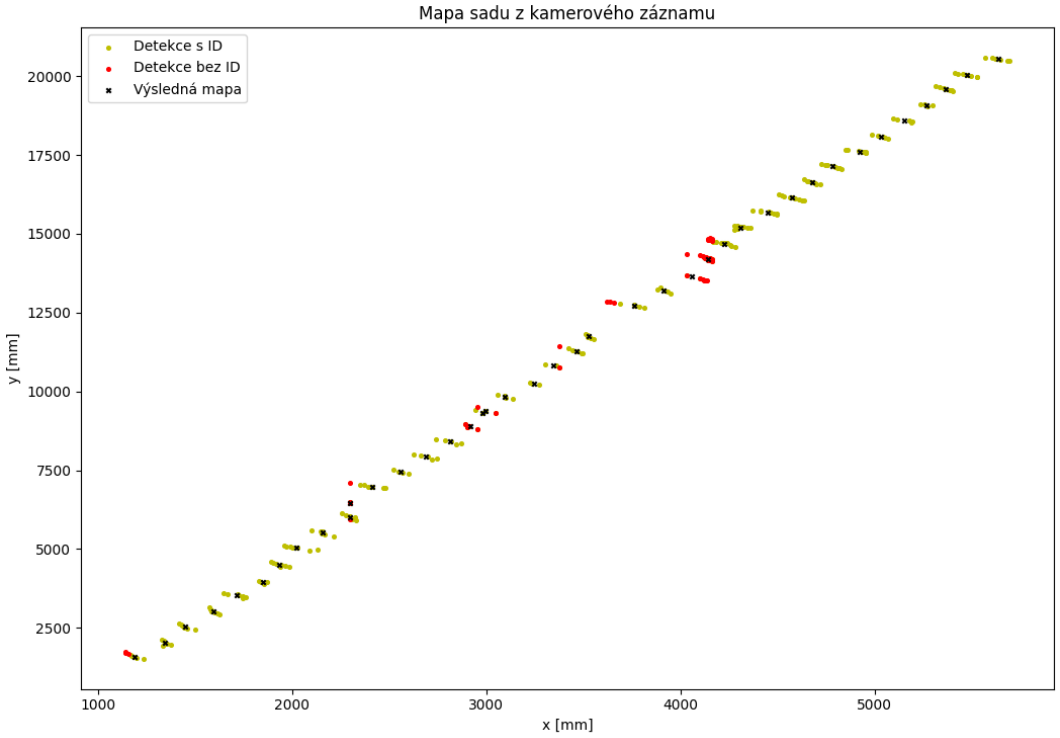
Tato část se podobně jako *lidarmap_cleaner* stará o zpracování detekovaných stromů a tvorbu výsledné mapy. Identifikace a sledování stromů na po sobě jdoucích snímcích je v tomto případě obstaráváno pouze YOLO v8 algoritmem. Pokud tento algoritmus pozná, že se jedná o stejný kmen jako na předchozím snímku, připne mu identifikátor. Pro tento účel je třeba mít dostatečně vysokou frekvenci snímků, aby si nacvičená síť zvládla vytvořit spojitosti mezi po sobě následujícími obrázky.

Úspěšnost detekce kmenů na jednotlivých snímcích v prezentovaných datech byla počítána jako suma úspěšných detekcí vydělená sumou celkových výskytů stromů a dosáhla cca 98 %. Mezi všemi úspěšnými detekcemi stromů se vyskytly i případy, kdy detektor nepoznal strom na po sobě jdoucích snímcích a nepřiřadil tak detekovanému boxu identifikátor. Poměr detekcí s přiřazeným identifikátorem vůči všem úspěšným detekcím odpovídá necelým 92 %. Celý systém pro detekci a sledování stromů má tedy úspěšnost minimálně 90 %. Tato úspěšnost se dá relativně snadně zvýšit dalším trénováním na řádně anotovaných snímcích. Spolu s vývojem projektu postupně narůstá také množství měření a výsledných fotografií. Dá se tedy s rozumnou pravděpodobností předpokládat, že ve finální fázi bude neuronová síť natrénována na mnohem vyšší procento úspěšnosti a sledování stromů v kamerovém obrazu

bude možné nechat pouze na ní. Pro zlepšení fungování v testovací a vývojové fázi, ve které se nacházíme byl nasazen podobný způsob, který byl využit u lidaru.

Tato část byla pro přehlednost ve zdrojovém kódu č. 7 vynechána. Jedná se o situaci, kdy je nalezen strom v definovaném intervalu bez přiděleného identifikátoru. V ten moment nastoupí část algoritmu, která predikuje pozici posledního uloženého stromu pomocí známého posunu robota. Pokud je vzdálenost predikovaného a detekovaného stromu větší, než je identifikační limit, jedná se o nový strom, kterému je připnut nový identifikátor. Aby nedošlo k narušení číslování zavedeného detektorem YOLO, začíná nové číslování od hodnoty 1000. Pokud je vzdálenost nižší, algoritmus vyhodnotí, že se jedná o stejný posunutý strom a připne mu odpovídající index. Takto doplněné sledování stromu na snímku dokáže správně zareagovat i v momentech, kdy snímaný strom na jednom snímku má přidělený identifikátor a na druhém ne. Výhodou tohoto způsobu doplnění identifikátorů je plynulé navázání na vytvořené číslování samotným detektorem a část kódu je tak spouštěna pouze v nutných případech.

Po programové stránce dochází pro každý identifikátor k průměrování odpovídajících měření, konkrétně souřadnic stromu a času. Na rozdíl od přístupu využitého v bloku *lidarmap_clean* se jedná o vážený průměr, kde váhy jsou udávány jednotlivými koeficienty důvěry ve správnost detekce. Souřadnice stromu určeného s jistotou 80 % má pak dvojnásobný vliv na výslednou pozici než souřadnice stromu určená s jistotou 40 %.



Graf 3: Mapa z kamerového záznamu

Graf č. 3 představuje vytvořenou mapu, kde žlutou barvou jsou vyneseny všechny detekce obsažené v mapě *cameramap_raw*, červenou všechny detekce bez identifikátoru přiděleného neuronovou sítí a černě výsledné souřadnice stromů. Hlavní výhodou tímto přístupem vytvořené mapy je rozeznání rozdílu mezi stromem a sloupem a mapa tedy obsahuje pouze souřadnice stromů.

4.3.5 Amap_maker, map_fusion a photo_extractor

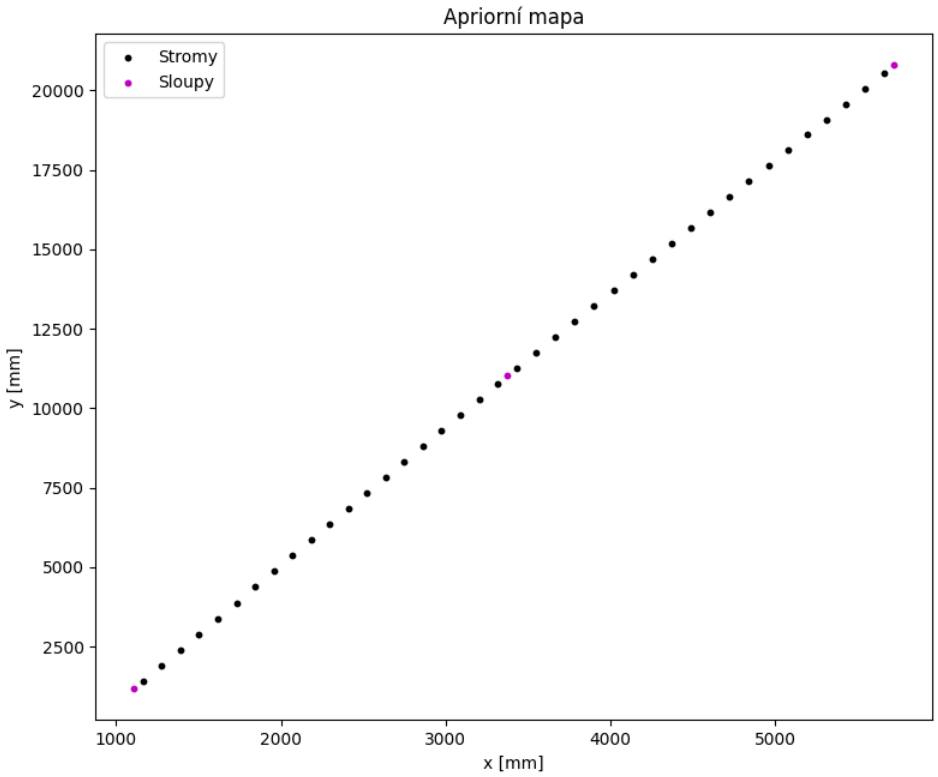
Tvorba apriorní mapy

Poslední ze tří později kombinovaných map je mapa vytvořená na základě apriorních informací. Tento blok může běžet paralelně s kamerovou detekcí, hned po tom, co obdrží informace o pozici robota z *fused_trajectory*. Apriorní informace před začátkem měření obsahují GPS souřadnice jednotlivých sloupů, počet stromů v poli (definovaná oblast mezi dvěma sloupy) a vzdálenost mezi sazenými stromy. Testovací stromová řada obsahuje celkem třináct po sobě jdoucích polí, na každém poli se nachází dvacet stromů se vzájemným rozpětím 50 cm. Konec a začátek každé řádky obsahují špatně detekovatelné ukotvení vodících lan. Pro autonomní nebo semiautonomní pohyb robota v sadu je potřeba tato ukotvení zaměřit a vynést do mapy. Pro apriorní mapu by teoreticky stačilo zaměřit pouze první a poslední strom v řadě a na základě známého počtu polí by byla pozice dalších sloupů odhadnuta. U tohoto přístupu, je ale velká šance kumulace malých odchylek a odhadnuté pozice stromů v jednotlivých polích by nemusely odpovídat realitě. Přístup, využitý v této práci, pracuje se známými GPS souřadnicemi všech sloupů v řadě, což značně redukuje možné odchylky.

```
def generate_trees(point_1, point_2, num_trees):
    x1, y1 = point_1
    x2, y2 = point_2
    pole_dist = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
    #Vzdalenost mezi jednotlivymi stromy
    spacing = pole_dist / num_trees
    print("spacing", spacing)
    unit_vector = ((x2 - x1)/pole_dist, (y2 - y1)/pole_dist)
    tree_coordinates = []
    for i in range(num_trees):
        #Vypocet pozice kazdeho individualniho stromu
        x_tree = x2 - (i+0.5) * spacing * unit_vector[0]
        y_tree = y2 - (i+0.5) * spacing * unit_vector[1]
        tree_coordinates.append([x_tree, y_tree])
    return tree_coordinates
```

Zdrojový kód 8: Funkce generování souřadnic stromů

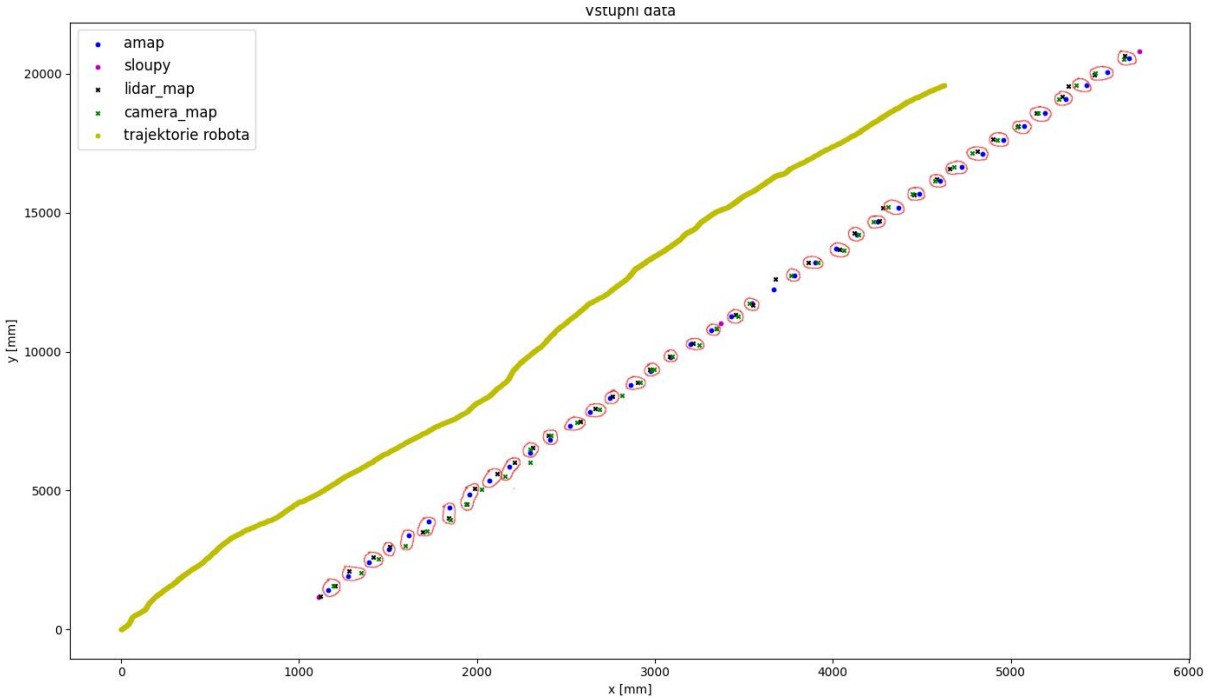
Stromy jsou generovány na základě vstupních parametrů, jimiž jsou souřadnice sloupů, mezi kterými se řada stromů nachází, a počet stromů, které chceme vygenerovat. V programové část se tímto úkolem zaobírá funkce *generate_trees(),* zobrazená ve zdrojovém kódu č. 8. Vstupem jsou soubory souřadnic ve formátu x, y. Převedení mezi souřadnicovými systémy zajišťuje funkce *gps2xy(),* která má za vstup počátek souřadnicového systému a souřadnice sloupu, který chceme převést. Kartézské souřadnice jsou jednoduše získány na základě rozdílů posunů v jednotlivých osách pomocí knihovny *geopy*. Mezi vstupními souřadnicemi sloupů je následně rovnoměrně rozmístěn počet stromů. Pro kontrolu každé použití funkce vypíše průměrnou vzdálenost stromů, kterou nadále můžeme porovnat s předem známým rozpětím stromů. Výstupem této části je *a_map*, obsahující souřadnice jednotlivých stromů a sloupů (graf č. 4)



Graf 4: Vytvořená apriorní mapa

4.3.5.1 Fúze mapových dat

Přístup tvorby výsledné mapy sadu v této práci byl kombinací map ze tří zdrojů, které na sobě nejsou přímo závislé. Hlavní důvod tvorby mapy je identifikace jednotlivých stromů a jejich uložení do databáze. Každý strom bude v databázi obsahovat soubor fotek z různých měření v různé roční doby. Takto vytvořené digitální dvojče sadu umožňuje sadařům ukládat a analyzovat historické informace o konkrétních stromech, na jejichž základě mohou provádět důležitá rozhodnutí.



Graf 5: Vstupní data pro mapovou fúzi s ručně vyznačenou detekcí stromů

Graf č. 5 představuje barevně odlišené vstupy jednotlivých detekovaných stromů. Modře jsou vyznačeny stromy apriorní mapy, černě stromy z lidarové mapy a zeleně stromy z kamerové mapy. Na obrázek byly pro zvýšení přehlednosti ručně červenou barvou vyznačeny odpovídající si shluky.

Zvolený způsob spojování map vychází z principu dva ze tří. Pokud je strom na dané pozici identifikován alespoň dvěma mapami, je považován za správně detekovaný a je vynesen do hlavní mapy. Jako reference pro porovnání a přiřazování odpovídajících si dat byla zvolena apriorní mapa. Tím, že mapa vytvořená z lidarových dat a mapa vytvořená z kamerových snímků při své tvorbě nepočítají s počtem stromů, které mají identifikovat, se může stát, že je identifikovaný jiný počet stromů, než by sadař v poli očekával. Informace o počtu stromů obsažených v apriorní mapě je tedy klíčová a můžeme jí tedy považovat za referenci.

```
for l in range(len(amap)):
    #Lidarova cast
    closest_l = 1000000
    closest_l_ind = 1000000
    #Ukladani nejkratsi vzdalenosti ke kontrolovanemu bodu
    for i in range(len(center_lidar)):
        dd = math.dist(amap[l], [lidarx[i], lidary[i]])
        if dd < closest_l:
            closest_l = dd
            closest_l_ind = i

    if closest_l > merge_limit:
        #Pokud je nejkratsi vzdalenost vetsi nez limit
        print("Neidentifikovano")
        #Pripnuti np.nan do listu souradnic a casu
        closest_list_lidar.append([np.nan, np.nan])
        closest_list_lidar_t.append([np.nan, np.nan])
        nonidentified_lidar.append([center_lidar[closest_l_ind][1][0],
                                    center_lidar[closest_l_ind][1][1]])
        nonidentified_lidar_t.append(center_lidar[closest_l_ind][0])

    else:
        #Pripnuti nejblizsiho centra a casu do listu
        closest_list_lidar.append([center_lidar[closest_l_ind][1][0],
                                   center_lidar[closest_l_ind][1][1]])
        closest_list_lidar_t.append(center_lidar[closest_l_ind][0])
```

Zdrojový kód 9: Hledání nejbližších stromů

Samotná fúze probíhá kontrolou každého bodu apriorní mapy a hledání nejbližšího detekovaného stromu v obou zbývajících mapách (zdrojový kód č. 9). Pokud je nalezen strom v bližší vzdálenosti, než je limitní vzdálenost daná parametrem *merge_limit* je připnut do listu, pokud ne, je do listu připnuta hodnota *np.nan*. Zároveň je vyšetřovaný bod připnutý do listu nepřiřazených bodů pro další kontrolu. Stejným způsobem, který je naznačený ve zdrojovém kódu č. 9, je řešená i část s mapou vytvořenou z kamerových dat.

Výstupem jsou listy *closest_list_arecont* a *closest_list_lidar*, které jsou v následující části porovnávány s hodnotami z *amap*. Jednotlivé prvky listů se stejnými indexy jsou porovnávány na základě dvou možných stavů – obsahují hodnotu nebo np.nan. Tabulka č. 5 zobrazuje možné kombinace, které dávají smysl a výsledek, který ze stavů jednotlivých způsobů vyplývá.

Tabulka 5: Možné kombinace vstupů

Lidar_map	Camera_map	Apriori_map	Výsledek
✅	✅	✅	✅
❌	✅	✅	✅
✅	❌	✅	✅
✅	✅	❌	⚠
❌	❌	✅	❌

První tři řádky popisuji případy, kdy byl strom identifikovaný apriorní mapou a zároveň alespoň jedním dalším způsobem, čímž byla existence stromu potvrzena. Kombinace na čtvrtém řádku tabulky podle definice také splňuje podmínku dva ze tří, ale apriorní mapa žádnou pozici stromu neregistruje. Výskyt souřadnic stromu v apriorní mapě je klíčový pro identifikaci, pokud tento případ nastane, výskyt stromu potvrzený nebude. Po dokončení hlavní smyčky jsou porovnány listy se souřadnicemi stromů z kamerové a lidarové mapy, které nebyly přiřazeny k žádným bodům apriorní mapy. Pokud jsou některé body v těchto listech blíže, než je hodnota parametru *merge_limit,* program vypíše varování o možném vynechání stromu s časem, ve kterém k tomuto případu došlo. Pokud se v jednom měření těchto varování vyskytne větší počet, může to znamenat, že apriorní mapa neposkytuje dostačující reprezentaci prostředí. Poslední řádek ošetřuje případ chybějícího stromu v řadě, kdy lidarová ani kamerová data nepotvrdí výskyt. Strom tímto způsobem identifikovaný není, ale informace o jeho absenci je předána do dalšího bloku *photo_extractor*, pro zachování správného číslování stromů.

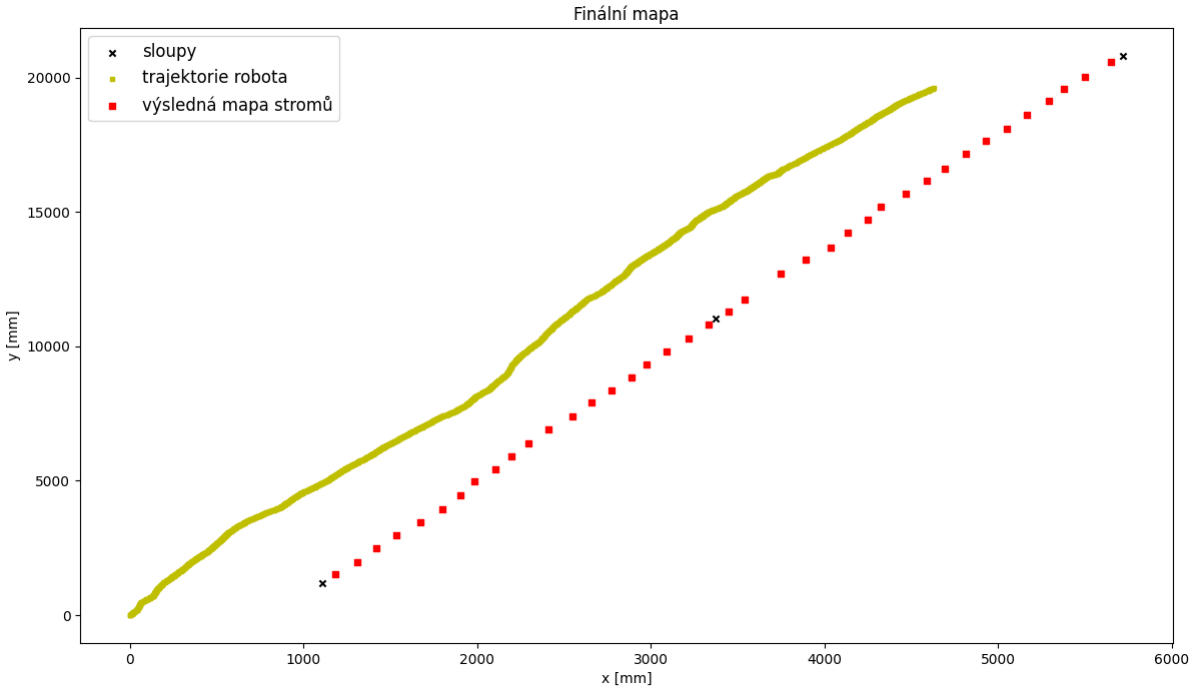
```
#Prochazeni kazdeho bodu apriorni mapy
for kk in range(len(amap)):
    cam_value = closest_list_arecont[kk]
    lid_value = closest_list_lidar[kk]
    amap_value = amap[kk]

    cam_time = closest_list_arecont_t[kk]
    lid_time = closest_list_lidar_t[kk]

    if not np.isnan(cam_value[0]) and not np.isnan(lid_value[0]) and not np.
    isnan(amap_value[0]):
        print("Identifikace vsemi tremi zpusoby")
        #Vazeny prumer odpovidajicich si bodu
        cent = [amap_value[0] * amap_weight + lid_value[0] * lidar_weight +
                cam_value[0] * camera_weight,
                amap_value[1] * amap_weight + lid_value[1] * lidar_weight +
                cam_value[1] * camera_weight]
        #Vazeny prumer casu
        timer = (cam_time * camera_weight + lid_time * lidar_weight)/
                (camera_weight + lidar_weight)
        #Pripnuti casu a pozice finalniho stromu
        fused_map.append([timer, cent])
```

Zdrojový kód 10: Smyčka porovnávající odpovídající si hodnoty map

Zdrojový kód č. 10 obsahující smyčku procházející všechny body apriorní mapy případ, ve kterém se na identifikaci stromu shodnou všechny tři způsoby. Výsledná souřadnice je pak dána váženým průměrem jednotlivých složek. Kromě souřadnice je zde průměrován i čas a připínán do konečného listu pro využití v bloku *photo_extractor*. Zbylé úspěšné případy uvedené v tabulce č. 5 jsou vyřešeny v kódu obdobně.



Graf 6: Výsledná mapa sadu

Výsledná mapa je zobrazena v grafu č. 6, kde jsou souřadnice výsledných stromů označeny červenými čtverci, sloupy černými křížky a žlutou barvou trajektorie robota. Výsledné váhy použité pro určení souřadnic stromů byly: váha apriorní mapy = 0,4, váha lidarové mapy = 0,3 a váha pro kamerovou mapu = 0,3. Toto nastavení dává podobnou váhu všem zdrojům informací o poloze a můžeme vidět, že výsledná množina stromů tak netvoří dokonalou přímku. Důležitý parametr *merge_limit* pro přiřazování bodů jednotlivých map měl hodnotu 20 cm, z čehož vychází, že každý strom je určený s přesností cca 40 cm. Vstupní data obsahovala chybějící strom na třetí pozici v druhém poli, což algoritmus správně rozpoznal. Výstupem tohoto bloku je *fused_map*, která obsahuje finální souřadnice stromů a časy, ve kterých byly stromy detekovány.

Zisk fotografií jednotlivých stromů

Vstupem do bloku *photo_extractor*, který se zabývá primárně ziskem výsledných fotografií stormů, jsou *fused_map*, natrénovaný model detekce a kamerová data pro zisk fotek stromů *arecont_cam* a *realsense_cam*. Kamerové snímky díky svému zornému úhlu a vzdálenosti, ve které jsou pořízeny, obsahují více stromů a je třeba nějakým způsobem detekovat ten správný. K tomu účelu slouží časový údaj připnutý ke každé souřadnici stromu. Pro všechny tyto časové údaje jsou nalezeny snímky jednotlivých kamer a na snímcích arecont_kamery je provedena detekce kmenů stromu. Následující část ve zdrojovém kódu č. 11 představuje řešení pro zisk oříznuté fotografie na základě úspěšností detekce. Na tímto způsobem pořízených fotografiích bývají stromy na obrázku v téměř neměnné oblasti. Pokud tedy není na obrázku detekován kmen bude pro ořez využitý předem definovaný vypozorovaný pixelový interval. Pokud na obrázku dojde k detekci kmene, je řešena jeho pozice. V případě, že žádný z detekovaných stromů neleží v pásmu ohraničeném parametrem *distortion_limit*, je opět použit výchozí pixelový interval pro ořez. V případě, že se nalezený kmen vyskytuje uvnitř intervalu, je použito oříznutí přizpůsobené pozici kmene. Od střední hodnoty ohraničujícího boxu je poté vynesena expertně určená fixní hodnota na obě strany a výsledné okno je použito pro ořez snímku. Tímto způsobem vzniknou fotky, které mají vždy kmen stromu uprostřed snímku.

```
if boxes is None:
    #Na snimku nebyl detekovan zadny strom - vychozi interval orezu
    frame_arecont = frame_arecont[:, default_crop_arecont[1][0]
                                    :default_crop_arecont[1][1]]
    frame_rscolor = frame_rscolor[:, default_crop_rscolor[1][0]
                                    :default_crop_rscolor[1][1]]
else:
    #Kontrola detekovanych boxu
    for m in range(len(boxes)):
        xbox = float(boxes[m][0]+boxes[m][2])/2
        #Vybirame si strom blizko stredu
        if xbox > distortion_limit and xbox < w_arecont - distortion_limit:
            fitcrop = True
            fitm = m
    if fitcrop:
        xbox = float(boxes[fitm][0]+boxes[fitm][2])/2
        #Byl nalezen strom blizko stredu - prizpusobeni intervalu orezu
        frame_arecont = frame_arecont[:, int(xbox)-halfcrop_arecont + 1
                                       :int(xbox)+halfcrop_arecont]
        frame_rscolor = frame_rscolor[:, int(a2rs(xbox, w_rscolor, w_arecont)) -
                                        halfcrop_rscolor + correction + 1:
                                        int(a2rs(xbox, w_rscolor, w_arecont)) +
                                        halfcrop_rscolor - correction]
    else:
        #Nebyl detekovan strom blizko stredu - vychozi interval orezu
        frame_arecont = frame_arecont[:, default_crop_arecont[1][0]
                                       :default_crop_arecont[1][1]]
        frame_rscolor = frame_rscolor[:, default_crop_rscolor[1][0]
                                       :default_crop_rscolor[1][1]]
```

Zdrojový kód 11: Přizpůsobení ořezávacího okna detekovanému kmenu

Šířky i výšky výsledných snímků jednotlivých stromů jsou dány zvolenými parametry. Díky sloupcovitému charakteru stromů dosahuje tato metoda dobrých výsledků. Alternativní přístup využitelný v momentě, kde jsou snímané stromy různorodé, nahnuté apod., by mohl využívat hloubkové kamery. Ta přiřazuje každému pixelu obrazu barvu na základě naměřené vzdálenosti, takže by se využitím dolno-propustního filtru mohl odhadnout tvar koruny a ořezávací okno by se pak dalo více přizpůsobit. Další analýza uložených snímků v této fázi nepožaduje specifické rozlišení. Pokud by někdy v budoucnosti projektu bylo potřeba tvořit snímky na míru každému stromu, dá se toho tímto způsobem dosáhnout.

Všechny kamery mají v současné instalaci objektivy natočeny stejným směrem a osy snímků zabírají vždy stejné místo. Pro detekci stromů na snímcích ze všech kamer stačí pouze, aby detekce fungovala spolehlivě na jednom druhu kamery. Na základě známých parametrů jako je zorný úhel a rozlišení lze určit odpovídající část jednoho snímku na druhém. V této verzi je na fotografii z Arecont kamery určen správný pixelový interval v horizontální ose, který je poté použit k nalezení odpovídajícího intervalu na snímku z RealSense kamery. V případě nízké frekvence snímkování by se mohlo stávat, že rozdíl mezi nejbližšími odpovídajícími si snímky jednotlivých kamer přesáhne limitní hodnotu a jednotlivé oblasti si nebudou stoprocentně odpovídat. V tom případě se dá využít závislosti mezi ujetou vzdáleností robota za daný čas a reálným rozměrem jednoho pixelu. Ořezávací okno se pak náležitě posune proti nebo ve směru jízdy.

Snímkům je pak možno přiřadit jméno podle zadání ve formátu číslo řady – číslo pole – číslo stromu. Z předchozího bloku zůstala zachována informace o tom, kolik stromů v poli chybí. Jelikož pohyb robota byl rovnoměrný a stromy jsou v poli rozmístěné také rovnoměrně, ve výsledných souřadnicích stromů se v případě chybějícího stromu objeví značné mezery. Funkce *gap_index* hledá v souřadnicích stromů počet největších mezer mezi sousedními prvky a vrátí indexy těchto mezer v listu. Pokud má být při číslování stromu přiřazen index, který se nachází v listu známých mezer, je tento index přeskočen a je přiřazen index vyšší.

Obrázek 34: Výsledný snímek stromu z Arecont kamery (vlevo) a RealSense kamery (vpravo)

Výslednou strukturou je list *tree_list* obsahující informace o dopočítaných GPS pozicích stromů a snímky jednotlivých stromů (obrázek č. 34). Pro další vývoj jsou do listu připnuty i informace o času, ve kterém byl strom v daném měření detekován a souřadnice v souřadnicovém systému specifickém pro dané měření.

4.3.6 Přístup pro rozsáhlá měření a zhodnocení metody

Měření, jehož výsledky byly prezentovány v předchozí části, probíhalo na předem vybraném rozpětí dvou polí s kapacitou čtyřiceti stromů. Použitý soubor metod správně identifikoval všechny stromy a jejich fotografie byly uloženy do databáze. Tato kapitola má za cíl zhodnotit efektivitu použitého přístupu pro rozsáhlejší vstupní data a navrhnout alternativy pro urychlení procesu. Vstupní data v tomto měření obsahovala jízdu robota kolem celé testovací řady o délce třinácti polí, kde každé pole obsahuje dvacet stromů. Grafická zpracování jednotlivých výstupů jsou do práce vloženy jako její přílohy. Programu byla ponechána jeho struktura a fungování popsané v předchozí fázi. Jedním z rozdílů bylo zvýšení váhy GPS měření pro tvorbu výsledné trajektorie, jelikož data uložená v kanálu pose3d obsahují kumulovanou odchylku natočení robota a tvar trajektorie neodpovídá rovnému pohybu mezi dvěma řadami.

Přílohy č. 1, 2 a 3 jsou alternativy grafů č. 2, 3 a 4 tentokrát pro delší vstupní data. Příloha č. 4 odpovídající grafu č. 5 zobrazuje jednotlivé mapy vynesené do jednoho grafu a příloha č. 5 podobně jako graf č. 6 zobrazuje výslednou mapu. Data obsahovala chybějící stromy, jejichž pozice byla správně určena a zohledněna při číslování. Algoritmus byl schopen správně detekovat 256 z 256 stromů a jejich oříznuté fotografie uložit do databáze. Nevýhodou představeného přístupu je značná časová náročnost zpracování dat. Délka běhu bloku *main_fusion*, ve které dochází k tvorbě mapy z lidarových dat a výsledné trajektorie, byla pro devítiminutové měření robotem necelých dvacet minut. Tvorba mapy z kamerových dat v bloku *camera_track* trvala pro stejná vstupní data sedm minut. Zdaleka největší dobu v těchto procesech zabírá zpracování lidarových dat. Pro každý cyklus jsou všechny naměřené body přepočítávány do kartézských souřadnic a následně shlukovány *mean-shift* algoritmem. Jedním z možných urychlení je předzpracování lidarových dat. Pro hledání vhodného zorného úhlu a maximální vzdálenosti byla tato část pro vývoj ponechána v hlavním programu. Posledním časově náročným blokem je *data_gathering*, který čte data uložená v záznamu a

ukládá je do datových struktur. Ostatní bloky včetně ukládání výsledných fotografií měly operační dobu pro stejná data v jednotkách nebo desítkách sekund.

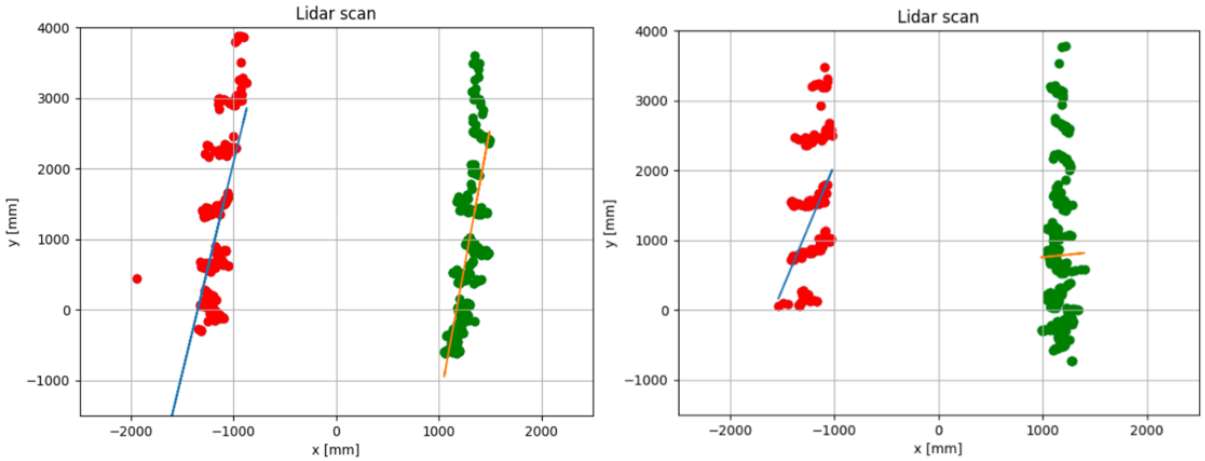
4.4 Režim poloautonomního snímkování

Tato část se zabývá přehledným poskytnutím informací obsluze robota o výsledcích a průběhu měření. Navržená aplikace nám pomůže odhalit problémy s jednotlivými informacemi ze senzorů rovnou při měření a je tak možné v případě špatných výsledků měření provést znova. V další části je popsán návrh autonomního řízení robota v rámci jedné řady. Původní vize shromažďování fotek stromů pracovala se zastavováním u každého stromu a následným statických focením. Od této vize bylo postupně upouštěno od pořízení Route kamery, která poskytuje ostré fotky i v dynamickém režimu. Testování bude provedeno na nové sadě dat, která nové fotografie z Route kamery obsahuje. Vstupní data budou simulovaně načítána z datových struktur, kam byla v bloku zaměřeném na sběr dat uložena. Výsledná implementace bude porovnána z výkonnostního a časového hlediska pro možnost uplatnění v live režimu. Kontrolovaná a vynášená data budou informace ze senzorů GPS a jednotlivých kamer. Lidarové snímky budou vyneseny a bude z nich pro operátora robota určena vzdálenost od jednotlivých stromových řad jako reference pro přímočarý pohyb.

4.4.1 Lidarová kontrola

V novém měření se kromě snímků z přidané kamery můžeme setkat i s pozměněným natočením lidaru ve směru cesty. Tato změna nám dává možnost monitorovat obě řádky současně a lépe detekovat objekty před vozidlem. Prioritou je zde namísto lokalizace jednotlivých stromů lokalizace jednotlivých řad a určení jejich vzdálenosti od senzoru lidaru. Můžeme si tedy dovolit odfiltrovat příliš krátké a příliš dlouhé naměřené vzdálenosti. Oproti minulému měření se také značně zvětšil analyzovaný zorný úhel na maximální hodnotu 270˚. Jelikož se jedná o pomocníka manuálního řízení, můžeme si dovolit neuvažovat dopravu robota na začátek řádky a řešíme tedy pouze jeho trajektorii v řádce. Díky známé vzdálenosti mezi sousedními řadami stromů známe přibližné pozice, kde by se řada měla nacházet. Druhý předpoklad o pohybu robota počítá s malými odchylkami od rovnoběžného pohybu, což vylučuje limitní případy, kdy je robot natočený ke stromové řadě pod úhly blízkými devadesáti stupňům. Dalším důsledkem tohoto předpokladu je rozdělení měřených intervalů na levou a pravou stranu. Pro dodatečnou kontrolu umístění senzoru mezi dvěma stromovými řadami slouží funkce *find_nearest_point(),* která určí nejkratší vzdálenost bodu predikujícího pozici stromové řady od listu obsahující všechny naměřené body. Pokud je tato vzdálenost menší než prahová hodnota na obou stranách, program předpokládá, že jsme opravdu mezi stromovými řadami.

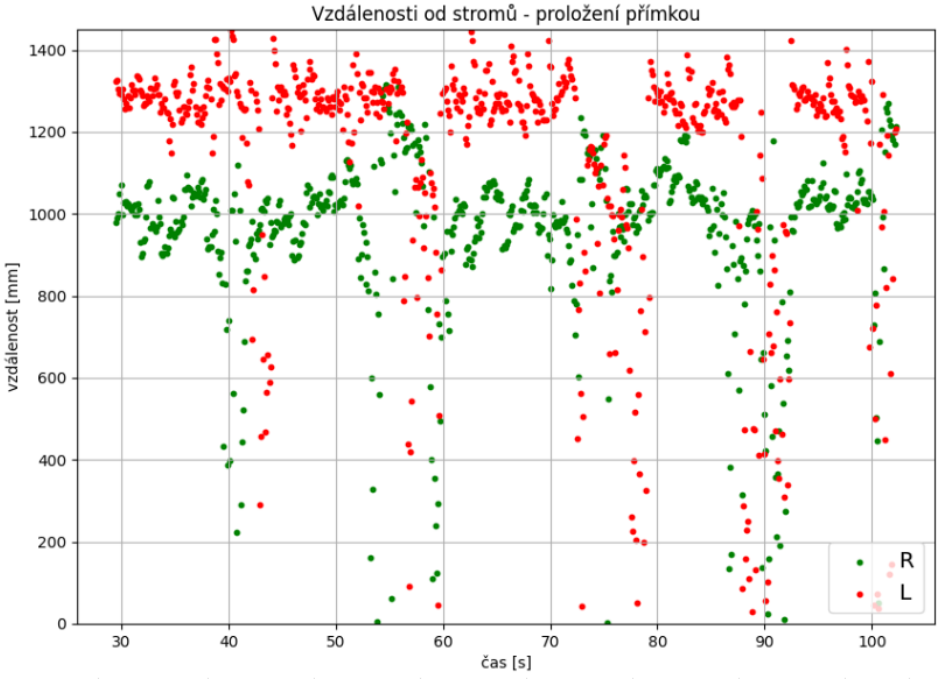
Vzdálenost robota od stromových řad je určena dvěma způsoby. První přístup určuje vzdálenost jako délku kolmice od senzoru na přímku proloženou stromovou řadou. Je zde využito rozdělení lidarových dat na dva intervaly odpovídající pravé a levé straně a každý z intervalů je analyzován zvlášť. Jelikož měření obsahuje i detekce ze vzdálenějších sousedních řad, omezíme se pro regresi na interval v okolí průměrné x-ové hodnoty souřadnic. Zvolený interval je dostatečně široký na to, aby odfiltroval vzdálená měření a zároveň nezahodil hodnoty vychýlené kvůli šikmému natočení řady. Na zbylé hodnoty v intervalu je nasazena metoda lineární regrese z knihovny *sklearn*. Výstupem této metody je sklon *a* proložené přímky. S pomocí pravidel analytické geometrie je určena rovnice proložené přímky a přímky, která je na ní kolmá a prochází středem souřadnicového systému, kde se lidarový senzor nachází. Výsledná vzdálenost je poté dopočtena z určeného průsečíku těchto přímek a jeho vzdálenosti od bodu [0, 0] reprezentujícího umístění senzoru.



Graf 7: Dva příklady pokusů o proložení lidarových dat přímkami

Graf č. 7 zobrazuje dva lidarové snímky, kde datová sada odpovídající pravé straně je vynesena zelenou barvou a levá strana má barvu červenou. Přímky vyneseny oranžovou a modrou barvou byly pro daná data určeny lineární regresí. Než budeme zkoumat natočení přímek, můžeme si všimnout, že shluky na pravé straně nejsou rozlišitelné jako na levé straně. Při tomto měření byla snaha lidar umístit co možno nejvýše pod hranici zavlažovací hadice. Kombinace malých nerovností terénu a nevodorovného připevnění lidaru způsobila, že u pravé stromové řady byla lidarem snímána oblast začátku koruny.

Levá strana obrázku zobrazuje většinový případ, kdy jsou přímky proloženy přijatelným způsobem. Problémem tohoto přístupu je nekonzistence proložení naznačená na pravé straně obrázku. Shluky blíže počátku souřadnicového systému obsahují mnohem více bodů než shluky vzdálené, tím pádem má model tendenci se více řídit bližšími shluky při výpočtu koeficientu sklonu.

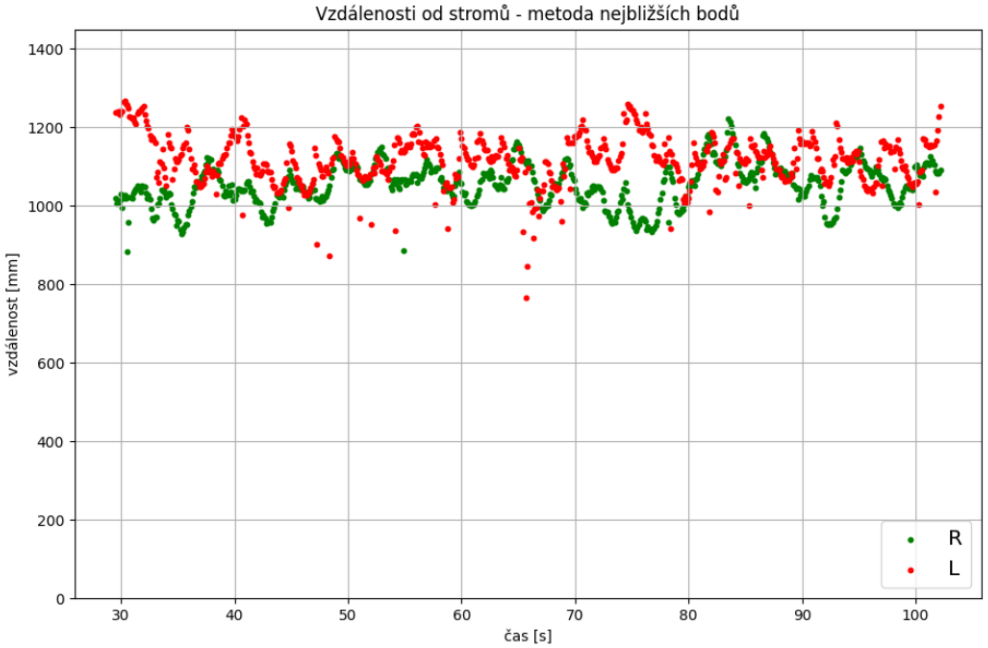


Graf 8: Závislost vzdálenosti na čase v režimu proložení přímkou

Graf č. 8 zobrazuje množiny vzdáleností počátku souřadnic od proložené přímky pro jednotlivé strany. Pro algoritmus řízení v linii na základě vzdálenosti od stromových řad jsou data získaná tímto způsobem obtížně použitelná. Jednou z testovaných alternativ bylo shlukování dat metodou *mean-shift* a následně provést regresi pouze na známých souřadnicích center shluků.

Tím došlo ale k výraznému zpomalení běhu aplikace a požadovaná konzistence nebyla dosažena. Více informací k těmto alternativám hledání řady stromů bude poskytnuto v kapitole zabývající se vedením robota v řádce.

Zvolený způsob určení vzdálenosti od stromové řady nakonec využívá pouze nejbližšího nalezeného bodu stromové řady. Tento přístup obsahuje odchylky v momentě, kdy v optimálním směru je mezera a nejkratší vzdálenost tak není kolmá na stromovou řadu. I přes tyto nedostatky jsou údaje o výsledné vzdálenosti od stromů konzistentní a použitelné pro případné řízení (graf č. 9).

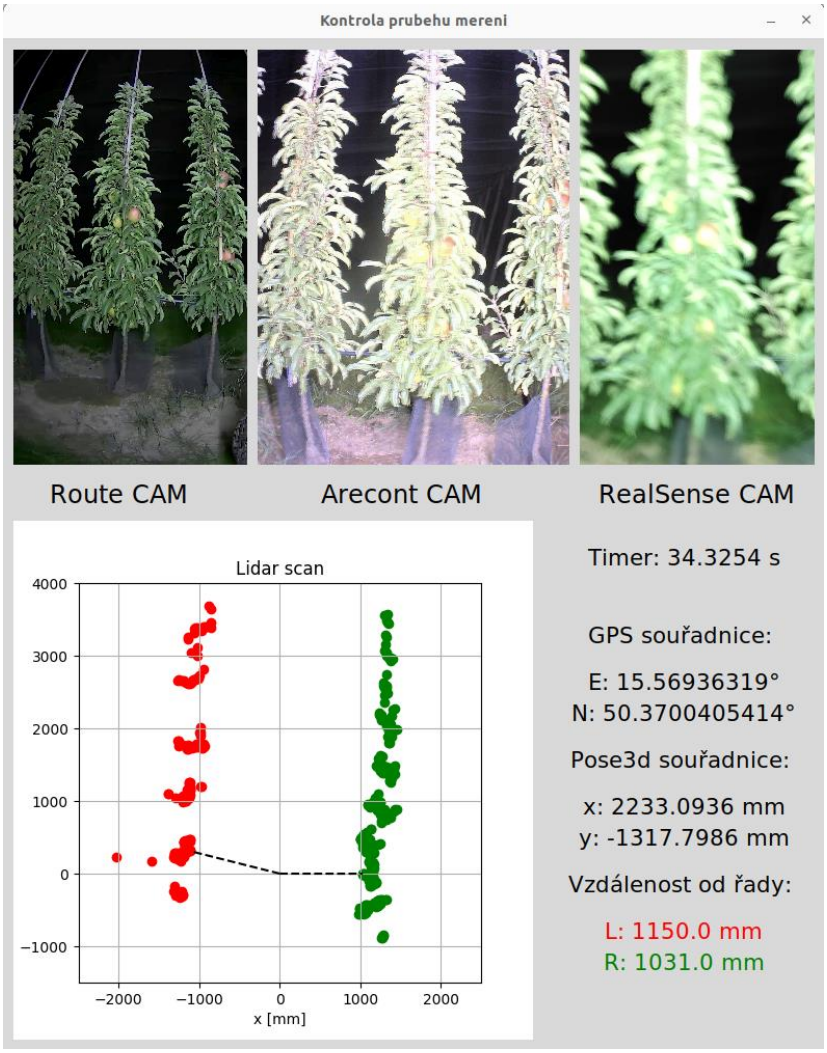


Graf 9: Závislost vzdálenosti na čase při využití nejbližšího bodu

4.4.2 Tvorba HMI

Pro tvorbu uživatelského rozhraní HMI (Human Machine Interface) je využita Python knihovna *tkinter*. Cílem je vytvořit jednoduchou přehlednou aplikaci zobrazující všechny potřebné věci, které by operátora robota mohly při jeho pohybu zajímat. Po programové stránce je uživatelské rozhraní tvořenou jednou třídou, které náleží tři specifické funkce. Inicializační funkce má za úkol spustit aplikaci, načíst jednotlivé datové z uložených listů, definovat potřebné proměnné a vytvořit okno včetně jeho grafických prvků. Aplikace vytvořená pomocí knihovny *tkinter* běžně funguje na základě hlavní stále se opakující smyčky. Po každém uběhnutí nastavené prodlevy se spustí funkce *update()* periodicky vykreslující a vypisující aktuální data a hodnoty do vytvořených grafických prvků.

Jednou z hlavních výzev je nalezení společného časového rámce, tak aby v jeden moment byla v aplikaci zobrazena data odpovídajícímu stejnému času. Při simulaci funkce na existujících uložených datech ve formě listů je tento úkol jednodušší. Při každém běhu aktualizační smyčky je zvýšen index určený k přístupu do jednotlivých listů. Tento index je pro každý informační zdroj vynásoben poměrem délky dat daného zdroje ku délce dat nejdelšího listu. Každá data jsou tedy zobrazována ve své frekvenci, se kterou byla uložena. Pro účely simulace můžeme přehrávání zrychlovat a zpomalovat nastavením prodlevy mezi každým cyklem. Pro spuštění aplikace v online režimu, by tento skript byl napojený jako modul na *OSGAR* a informace by byly čteny přímo z komunikačního kanálu. Bylo by také potřeba vytvořit další funkci, která by se starala o časovou synchronizaci a ukládala do proměnných odpovídající data. Tato data by z proměnných byla periodicky čtena a zpracovávána v existujících funkcích.



Obrázek 35: Uživatelské rozhraní aplikace

Zvolená struktura uživatelského rozhraní zobrazená na obrázku č. 35 obsahuje v horní části fotografie z jednotlivých kamer, pod nimi je vykreslen lidarový snímek a vedle něj jsou v pravém sloupci vypisovány ostatní údaje. Kamerové snímky jsou načítány z numpy polí, příslušně rotovány a jsou jim měněny rozměry při zachování stejných poměrů stran. Následně jsou v každém cyklu vykreslovány na své pozice na plátně. GPS souřadnice, informace o pozici z odometrie a časovač jsou pouze načítány z listů a následně vypisovány do příslušných textových polí. Lidarová data jsou v každém snímku přepočítávána z polárních souřadnic do kartézských, zapisována do listů a následně vykreslována. Spolu s detekovanými body jsou černě vykreslovány i čárkované úsečky označující nejkratší vzdálenost od obou řad.

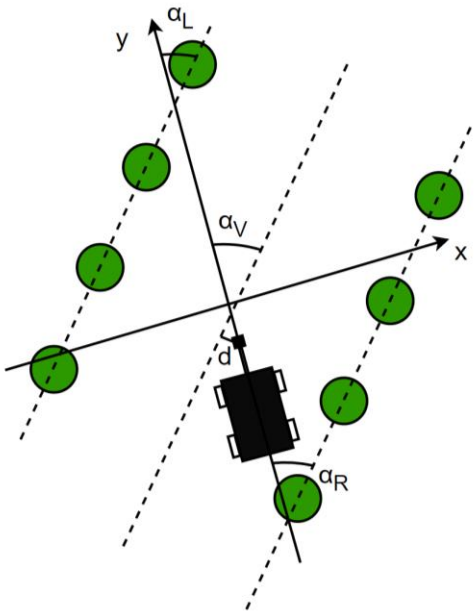
Hlavním cílem aplikace byla názornost a přehlednost, pro praktické využití na živých datech by možná bylo potřeba udělat několik změn. Současný běh aplikace není nijak optimalizovaný, a i tak zvládá na testovaném zařízení střední třídy zpracování a vykreslování dat rychleji, než je reálný běh času. Je potřeba dodat, že při spuštění na palubním počítači je potřeba počítat s pomalejším zpracováním dat. Kromě snížení frekvence obnovování dat je možné v případě potřeby namísto vykreslování plných obrázků pouze poskytnout údaj o tom, že kamera posílá data. Další možná změna je vykreslování snímku pouze z jedné kamery najednou a přepínat mezi okny v případě potřeby. Video z běhu aplikace je jednou ze součástí příloh práce.

4.4.3 Vedení robota v řádce

Tato kapitola volně navazuje na problém představený v kapitole zabývající se lidarovou kontrolou. Cílem je spolehlivě detekovat stromovou řadu a navrhnout možný přístup k jejímu

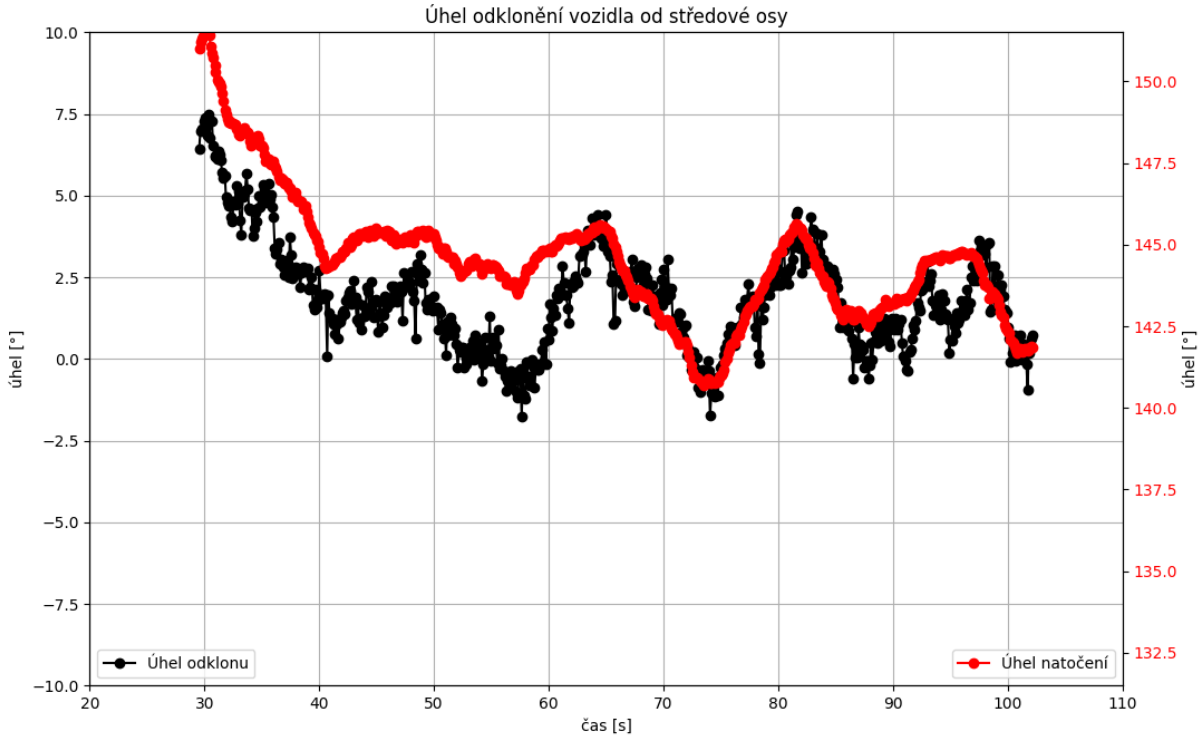
autonomnímu sledování. Doposud představené přístupy popisovaly uplatňování metody lineární regrese na všechny detekované body stromové řady nebo na soubor shluků určených metodou *mean-shift*. První přístup nebyl úspěšný, jelikož vstupní data nebyla rovnoměrně rozmístěná po zkoumaném intervalu a prokládání přímky souborem bodů tedy více upřednostňovalo oblasti blíže robota. Nevýhodou druhého přístupu využívajícího shluků je výpočetní náročnost shlukovacího procesu a následného procházení center. Zároveň jedno špatně identifikované nebo vybočené centrum radikálně ovlivní sklon a umístění prokládané přímky.

Jelikož v této části předpokládáme pohyb robota v rovné řádce, můžeme na základě známých rozměrů řádky odhadnout, kde by se konec zobrazované části řádky měl nacházet. Pokud je vzdálenost nejbližšího bodu z řady k odhadnutému bodu dostatečně nízká můžeme usoudit, že se jedná o bod dané řady. Pro zvýšení spolehlivosti můžeme nalézt soubor nejbližších bodů, ze kterých je výsledný bod určen, popř. opakovat proces na více místech řady. Získané body jsou poté proloženy přímkou, jejíž sklon je klíčový pro další navigaci.



Obrázek 36: Schéma pohybu robota v řádce

Tímto způsobem získané sklony obou přímek prokládajících pravou a levou stromovou řadu mohou být použity k tvorbě referenční přímky vedoucí buď přímo středem řádky nebo v definovaném posunu od něj. Schéma na obrázku č. 36 zobrazuje dvě přímky nakloněné o úhly α_L, α_R vůči souřadnicovému systému robota. Dvě důležité hodnoty pro navigaci robota jsou úhel naklonění referenční přímky α_V, a vzdálenost robota od přímky *d*. Řídicí systém *OSGAR* obsahuje modul, který umožňuje pohyb robota pod zadaným úhlem. Cílem řízení by v tomto případě byla minimalizace vzdálenosti *d* pomocí akční veličiny, kterou je úhel natočení robota.



Graf 10: Průběh úhlu odklonění vozidla od středové osy a úhlu natočení

Graf č. 10 popisuje černou barvou průběh úhlu odklonění vozidla od referenční osy získaný ze simulace zobrazené v příloze č. 7 a červenou barvou úhel natočení vozidla uložený v kanálu *pose3d*. Jeho cílem je demonstrovat spolehlivost určení úhlu odklonu daného referenční přímkou ve středu řádky. Ten kopíruje průběh skutečného úhlu natočení robota určeného gyroskopem s maximální odchylkou menší než 5°.

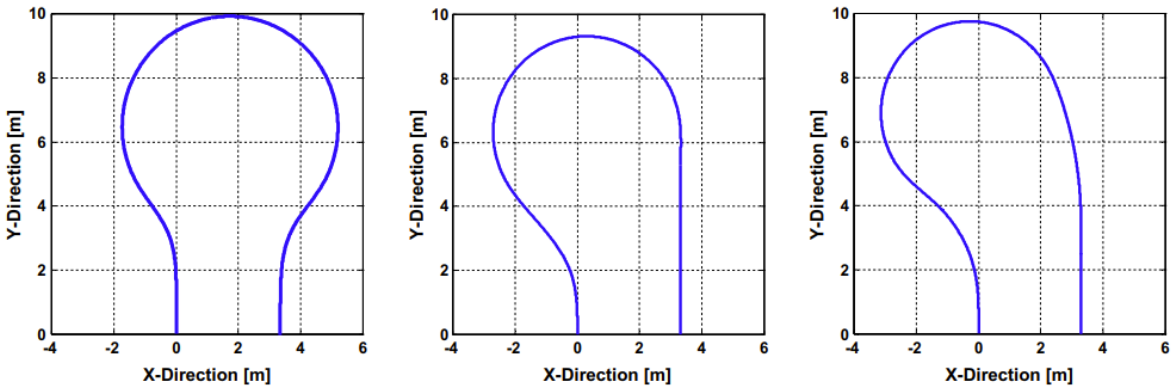
4.4.4 Otočka a zastavování

Robot je v navrhovaném systému navigován referenční přímkou. Narozdíl od manuálního řízení je při autonomním sledování řady potřeba sledovat oblast před robotem pro případné nouzové zastavení. Rychlost pohybu robota je regulována PID regulátorem, což s sebou přináší prodlevu přibližně 1 s mezi vysláním signálu pro zastavení a skutečným zastavením. Je tedy potřeba hlásit překážky v cestě s dostatečným předstihem. V prvotní fázi dává smysl při překročení kritické vzdálenosti předmětu v cestě vozidla zastavit i vzhledem k rozměrům robota, šířce řádky a doporučené vzdálenosti kamer od stromů.

V případě, že se robot v měření blíží ke konci řádky, což jsme schopni poznat podle vzdálenosti vozidla od známé souřadnice posledního sloupu, přechází robot do jiného režimu. Stromová řada už dále nepokračuje a prokládání přímkou je s každým dalším pohybem kupředu více nepřesné. V moment překročení definované vzdálenosti od posledního sloupu se tedy přestává obnovovat referenční přímka a zůstává stejná po celý zbytek řádky. Zároveň se zpřísní lidarová kontrola rozšířením intervalu kontroly objektů v cestě vozidla. Souřadnice konce řádky definovaného upnutými napínacími lany jsou známé kvůli špatné detekovatelnosti lidarem. V momentě, kdy bude ze vzdálenosti vozidla a upnutí lan zjištěno, že robot úspěšně projel kolem a lidar nedetekuje žádné překážky v cestě přichází na řadu otočka robota pro zajetí do sousední řádky.

Touto problematikou se zabývá článek *Localization and control of an autonomous orchard vehicle* [54], který navrhuje příklady tří možných trajektorií robota (obrázek č. 37) respektujících prostor, který má robot k otočení, bezpečnostní vzdálenost pro začátek otáčení, šířku řádky a v neposlední řádě minimální rádius zatáčení. Po celou dobu pohybu po

nadefinované křivce je třeba sledovat oblast před robotem pro případné zastavení v případě hrozící kolize. Po teoretické stránce by robot měl být schopen po zajetí do řádky opět spustit modul pro detekci stromových řad a dále navigovat v další řádce. Potenciálním spouštěčem by mohla být detekce QR kódu, připevněného na začátku řádky obsahujícího informace o stromech v řádce.



Obrázek 37: Možnosti zatáčení pro zajetí do nové řádky [54]

Část poloautonomního režimu a jeho autonomních prvků byla v této práci provedena pouze návrhově. Bylo zde představeno několik způsob detekce stromových linií, otestovaných na reálných datech z měření. Navržené prvky autonomního řízení se plně nepodařilo otestovat a jejich off-line simulace je při napodobení reálného prostředí obtížná. Aplikace navržených postupů na reálná data byla alespoň při práci na tomto projektu vždy ve skutečnosti komplikovanější a bylo třeba ošetřit některé podmínky, které nebyly v prvotním návrhu uvažované.

Závěr a zhodnocení

Cílem práce bylo představit integraci nových technologií do zemědělského prostředí se zaměřením na fúzi senzorů zahradního robota a tvorbu robotické mapy sadu. V rámci této práce vznikla rozsáhlejší rešerše s využitím velkého množství zahraničních zdrojů. Překlady ustálených frází a názvů jsou často doplněny o původní anglický termín pro snadné dohledání a jednoznačnost. Prvním probíraným tématem byla koncepce Zemědělství 4.0 od vzniku senzorické informace přes její předávání, zpracovávání, ukládání až po vizualizaci pro koncového uživatele. Jednotlivé využívané technologie byly stručně popsány po teoretické stránce a doplněny o příklady využití. Dalším tématem byla fúze senzorů zaměřená na její obecný popis, potíže a problémy s její praktickou aplikací. Jednotlivé fúzní metody byly rozděleny, stručně popsány a vzájemně porovnány co se charakteristik a využití týče. Poslední téma teoretické části je teorie robotického mapování, jeho úlohy a výzvy. Hlavní důraz je dbán na principy a algoritmy spojené se současnou lokalizací a mapováním a na popis a představení jednotlivých robotických map.

Praktická část je zaměřena na plnění zadaných úkolů v rozsáhlejším projektu zabývajícím se robotizací a digitalizací ovocného sadu. V rámci práce byl vyvinut program v prostředí Python, zaměřený na tvorbu robotické mapy ze senzorických dat. Dílčí mapy jsou vytvořeny třemi způsoby, které na sobě nejsou přímo závislé. První způsob hledá kmeny stromů na lidarových měřeních jakožto centra detekovaných shluků, druhý způsob zpracovává obrazová data pomocí dříve natrénovaného detektoru YOLO v8 detekujícího kmeny a třetí využívá známých souřadnic vodících sloupů na konci každého pole a informací o počtu stromů v poli. Jejich vhodným spojením vznikla finální mapa obsahující kromě souřadnic stromů i jejich oříznuté snímky pro další zpracování. Program byl ponechán ve své návrhové formě pro snadnější kontrolu mezivýsledků jednotlivých bloků a další vývoj.

Jednalo se o můj první větší Python projekt, u kterého byl hlavní důraz na funkčnost, snadné ladění pomocí parametrů, meziukládání a vizualizaci mezivýsledků. Jelikož se jedná o off-line zpracování dat, není první prioritou optimalizace, co se výkonu a trvání úkolů týče. Často byly nově vzniklé problémy vyřešeny pomocí přístupů na míru naší aplikaci a předpokladech o vstupních datech. Po programové stránce je tedy určitě místo ke zlepšení, především co se formální stránky týče. Každá individuálně vytvořená mapa obsahuje své nedostatky a odchylky spojené s měřením či metodou jejího získávání. Jejich kombinací byla vytvořena mapa správně identifikující všechny stromy s dostatečnou přesností na testovacích i rozsáhlejších vstupních datech. Jedním z největších nedostatků vstupních dat byla data získaná odometrií uložená v kanálu *pose3d*, která špatně reflektovala skutečnou trajektorii robota. V případě nových vstupních dat jsou v programu popsány části, které je nutné zkontrolovat a případná nastavení a vliv parametrů na výsledky jsou popsány v této práci nebo v přiloženém textovém souboru.

Druhý úkol praktické části se zaměřuje na návrh režimu poloautonomního snímkování a autonomních přístupů. Hlavní záměrem bylo přehledné poskytnutí klíčových informací operátorovi vozidla demonstrované tvorbou uživatelského rozhraní na základě dat uložených v záznamu. U navrženého uživatelského rozhraní byla zhodnocena jeho operační rychlost vykreslování informací, která v simulovaném prostředí operovala dostatečně rychle. Zároveň byly konzultovány možné modifikace výsledného programu pro zapojení do live-režimu a spouštění na méně výkonných zařízeních. Práce byla doplněna o možné přístupy k autonomnímu vedení robota v řádce převážně na základě lidarových měření a GPS. Na uložených datech byly představeny, testovány a zhodnoceny některé způsoby detekce stromové řady. Získané přímky vhodně proložené lidarovými daty sloužily k návrhu řízení na základě minimalizace vzdálenosti od referenční přímky pomocí změny úhlu natočení robota.

Na závěr práce byl představen možný přístup k přechodu z řádky do řádky. Bohužel se nepovedlo plně otestovat všechny metody na skutečném zařízení v reálném čase, tato část tedy působí převážně jako návrh pro další vývoj.

Reference

[1] ARAÚJO, Sara Oleiro, et al. Characterising the agriculture 4.0 landscape—emerging trends, challenges and opportunities. *Agronomy*. 2021, 37 s.

[2] ZHAI, Zhaoyu, et al. Decision support systems for agriculture 4.0: Survey and challenges. *Computers and Electronics in Agriculture*. 2020, 16 s.

[3] KOVÁCS, I.; HUSTI, I. The role of digitalization in the agricultural 4.0–how to connect the industry 4.0 to agriculture?. In: *Hungarian agricultural engineering*. 2018, s. 38–42.

[4] KUMAR, Ramakant, et al. Smart sensing for agriculture: Applications, advancements, and challenges. *IEEE Consumer Electronics Magazine*. 2021, s. 51–56.

[5] ULLO, Silvia Liberata; SINHA, Ganesh Ram. Advances in IoT and smart sensors for remote sensing and agriculture applications. *Remote Sensing*. 2021, 13.13: 2585. 14 s.

[6] Palanisamy, Shanmugapriya & Selvaraj, Rathika & Ramesh, Thanakkan & Ponnusamy, Janaki. Applications of Remote Sensing in Agriculture – A Review. *International Journal of Current Microbiology and Applied Sciences.* 2019, s. 2270–2283.

[7] RAHAMAN, Md Mohinur; AZHARUDDIN, Md. Wireless sensor networks in agriculture through machine learning: A survey. *Computers and Electronics in Agriculture*. 2022, 23 s.

[8] KHUJAMATOV, Kh E.; TOSHTEMIROV, T. K. Wireless sensor networks based Agriculture 4.0: challenges and apportions. In: *2020 international conference on information science and communications technologies (ICISCT)*. IEEE, 2020, s. 1–5.

[9] MARINOUDI, Vasso, et al. Robotics and labour in agriculture. A context consideration. *Biosystems Engineering*. 2019, 184: s. 111–121.

[10] LYTRIDIS, Chris, et al. An overview of cooperative robotics in agriculture. *Agronomy*. 2021, 23 s.

[11] BERENSTEIN, Ron; EDAN, Yael. Human-robot collaborative site-specific sprayer. *Journal of Field Robotics*. 2017, s. 1519–1530.

[12] DEL CERRO, Jaime, et al. Unmanned aerial vehicles in agriculture: A survey. *Agronomy*. 2021, 19 s.

[13] AGCO. MARS: Robot system for planting and accurate documentation. *fendt.com* [online]. 2017 [cit. 21. 10. 2023]. Dostupné z: https://www.fendt.com/int/fendt-mars.

[14] BLENDER, Timo, et al. Managing a mobile agricultural robot swarm for a seeding task. In: *IECON 2016 – 42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2016, s. 6879–6886.

[15] CONESA-MUÑOZ, Jesús, et al. A multi-robot sense-act approach to lead to a proper acting in environmental incidents. *Sensors*. 2016, 19 s.

[16] VU, Quyen, et al. Trends in development of UAV-UGV cooperation approaches in precision agriculture. In: *Interactive Collaborative Robotics: Third International Conference, ICR 2018, Leipzig, Germany, September 18–22, 2018, Proceedings 3*. Springer International Publishing, 2018, s. 213–221.

[17] SARABU, Hemanth; AHLIN, Konrad; HU, Ai-Ping. Graph-based cooperative robot path planning in agricultural environments. In: *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2019, s. 519–525.

[18] TZOUNIS, Antonis, et al. Internet of Things in agriculture, recent advances and future challenges. *Biosystems engineering*. 2017, s. 31–48.

[19] MISHRA, Kanderp Narayan; KUMAR, Shishir; PATEL, Nileshkumar R. Survey on Internet of Things and its Application in Agriculture. *Journal of Physics: Conference Series*. IOP Publishing, 2021, s. 1–9.

[20] GORAYA, Major Singh; KAUR, Harjinder. Cloud computing in agriculture. *HCTL Open International Journal of Technology Innovations and Research (IJTIR)*. 2015, s. 1–5.

[21] EVSTATIEV, B. I.; GABROVSKA-EVSTATIEVA, K. G. A review on the methods for big data analysis in agriculture. In: *IOP Conference Series: Materials Science and Engineering*. IOP Publishing. 2021. s. 1–7.

[22] BENOS, Lefteris, et al. Machine learning in agriculture: A comprehensive updated review. *Sensors*. 2021, s. 1–55.

[23] ARA, Iffat, et al. Application, adoption and opportunities for improving decision support systems in irrigated agriculture: A review. *Agricultural Water Management*. 2021, 16 s.

[24] NAVARRO-HELLÍN, Honorio, et al. A decision support system for managing irrigation in agriculture. *Computers and Electronics in Agriculture*. 2016, s.121–131.

[25] KADIYALA, M. D. M., et al. An integrated crop model and GIS decision support system for assisting agronomic decision making under climate change. *Science of the Total Environment*. 2015, s. 123–134.

[26] The Electrical Engineering and Applied Signal Processing Series. *Handbook of Multisensor Data Fusion: Theory and Practice*. Edited by M. E. Liggins et al. CRC Press, Taylor & Francis Group, 2009. 872 s. ISBN: 978-1-4200-5308-1.

[27] FUNG, Man Lok; CHEN, Michael ZQ; CHEN, Yong Hua. Sensor fusion: A review of methods and applications. In: *2017 29th Chinese Control And Decision Conference (CCDC)*. IEEE, 2017, s. 3853–3860.

[28] FAYYAD, Jamil, et al. Deep learning sensor fusion for autonomous vehicle perception and localization: A review. *Sensors*. 2020, 35 s.
[29] YEONG, De Jong, et al. Sensor and sensor fusion technology in autonomous vehicles: A review. *Sensors*. 2021, 37 s.
[30] JULIER, Simon J.; UHLMANN, Jeffrey K. Using covariance intersection for SLAM. *Robotics and Autonomous Systems*. 2007, s. 3–20.
[31] CASTANEDO, Federico, et al. A review of data fusion techniques. *The scientific world journal*, 2013, 20 s.
[32] MUMTAZ, Aqeel; MAJID, Abdul; MUMTAZ, Adeel. Genetic algorithms and its application to image fusion. In: *2008 4th International Conference on Emerging Technologies*. IEEE, 2008, s. 6-10.
[33] KUEVIAKOE, Kangni, et al. Localization of a vehicle: A dynamic interval constraint satisfaction problem-based approach. *Journal of Sensors.* 2018, s. 1–12.
[34] STACHNISS Cyril, et al. Robotic Mapping and Exploration. Springer-Verlag Berlin Heidelberg, 2009. 203 s. ISBN 978-3-642-01096-5.
[35] THRUN, S. Robotic Mapping: A Survey. *Exploring Artificial Intelligence in the New Millenium/Morgan Kaufmann google scholar.* 2002, s. 237–267.
[36] SAEEDI, Sajad, et al. Multiple-robot simultaneous localization and mapping: A review. *Journal of Field Robotics*. 2016, s. 3–46.
[37] RACINSKIS, Peteris; ARENTS, Janis; GREITANS, Modris. Constructing Maps for Autonomous Robotics: An Introductory Conceptual Overview. *Electronics*. 2023, 22 s.
[38] JUNG, Jin-Woo, et al. Mobile robot path planning using a laser range finder for environments with transparent obstacles. *Applied Sciences*. 2020, 23 s.
[39] LOZENGUEZ, Guillaume, et al. Punctual versus continuous auction coordination for multi-robot and multi-task topological navigation. *Autonomous Robots*. 2016, s. 599–613.
[40] DENG, Wenbang, et al. Semantic RGB-D SLAM for rescue robot navigation. In: *IEEE Access.* 2020, s. 221320–221329.
[41] OVALLE-MAGALLANES, Emmanuel, et al. Transfer learning for humanoid robot appearance-based localization in a visual map. In: *IEEE Access*. 2021, s. 6868–6877.
[42] KOSTAVELIS, Ioannis; GASTERATOS, Antonios. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*. 2015, s. 86–103.
[43] XUEXI, Zhang, et al. SLAM algorithm analysis of mobile robot based on lidar. In: *2019 Chinese Control Conference (CCC)*. IEEE, 2019, s. 4739–4745.
[44] LI, Peng, et al. Evaluation of the ICP algorithm in 3D point cloud registration. In: *IEEE Access*. 2020, s. 68030–68048.
[45] TAKETOMI, Takafumi; UCHIYAMA, Hideaki; IKEDA, Sei. Visual SLAM algorithms: A survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*. 2017, 11 s.
[46] MACARIO BARROS, Andréa, et al. A comprehensive survey of visual slam algorithms. *Robotics*. 2022, 27 s.
[47] FUENTES-PACHECO, Jorge; RUIZ-ASCENCIO, José; RENDÓN-MANCHA, Juan Manuel. Visual simultaneous localization and mapping: a survey. *Artificial intelligence review*. 2015, s. 55–81.
[48] e-noc Systems. Sony Starvis IMX462 Full HD GigE Camera. *e-consystems.com* [online]. 2024 [cit. 12. 11. 2023]. https://www.e-consystems.com/gige-cameras/sony-starvis-imx462-full-hd-camera.asp
[49] AV Costar. MegaVideo Compact Series: AV3236DN. *sales.arecontvision.com* [online]. 2024 [cit. 12: 11. 2023]. Dostupné z: https://sales.arecontvision.com/product/MegaVideo+Compact+Series/AV3236DN
[50] Intel Corporation. Depth Camera D455. *store.intelrealsense.com* [online]. 2024 [cit. 12. 11. 2023]. Dostupné z: https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d455.html
[51] SICK AG. TiM571-2050101. *sick.com* [online]. 2024 [cit. 12. 11. 2023]. Dostupné z: https://www.sick.com/cz/en/catalog/products/lidar-and-radar-sensors/lidar-sensors/tim/tim571-2050101/p/p412444
[52] Robotika.cz. Osgar. *github.com* [online]. 2023 [cit. 1. 12. 2023]. Dostupné z: https://github.com/robotika/osgar.
[53] ZHANG, Shuo, et al. Research on 2D laser automatic navigation control for standardized orchard. *Applied Sciences*. 2020, 19 s.
[54] BAYAR, Gokhan, et al. Localization and control of an autonomous orchard vehicle. *Computers and Electronics in Agriculture*. 2015, s. 118–128.

Seznam obrázků a tabulek

Pozn.: Zdroje obrázků jsou citovány. V případě chybějící citace se jedná o obrázky vytvořené autorem této práce.

Přílohy

1 Mapa z lidarových dat pro rozsáhlé měření

2 Mapa z kamerového záznamu pro rozsáhlé měření

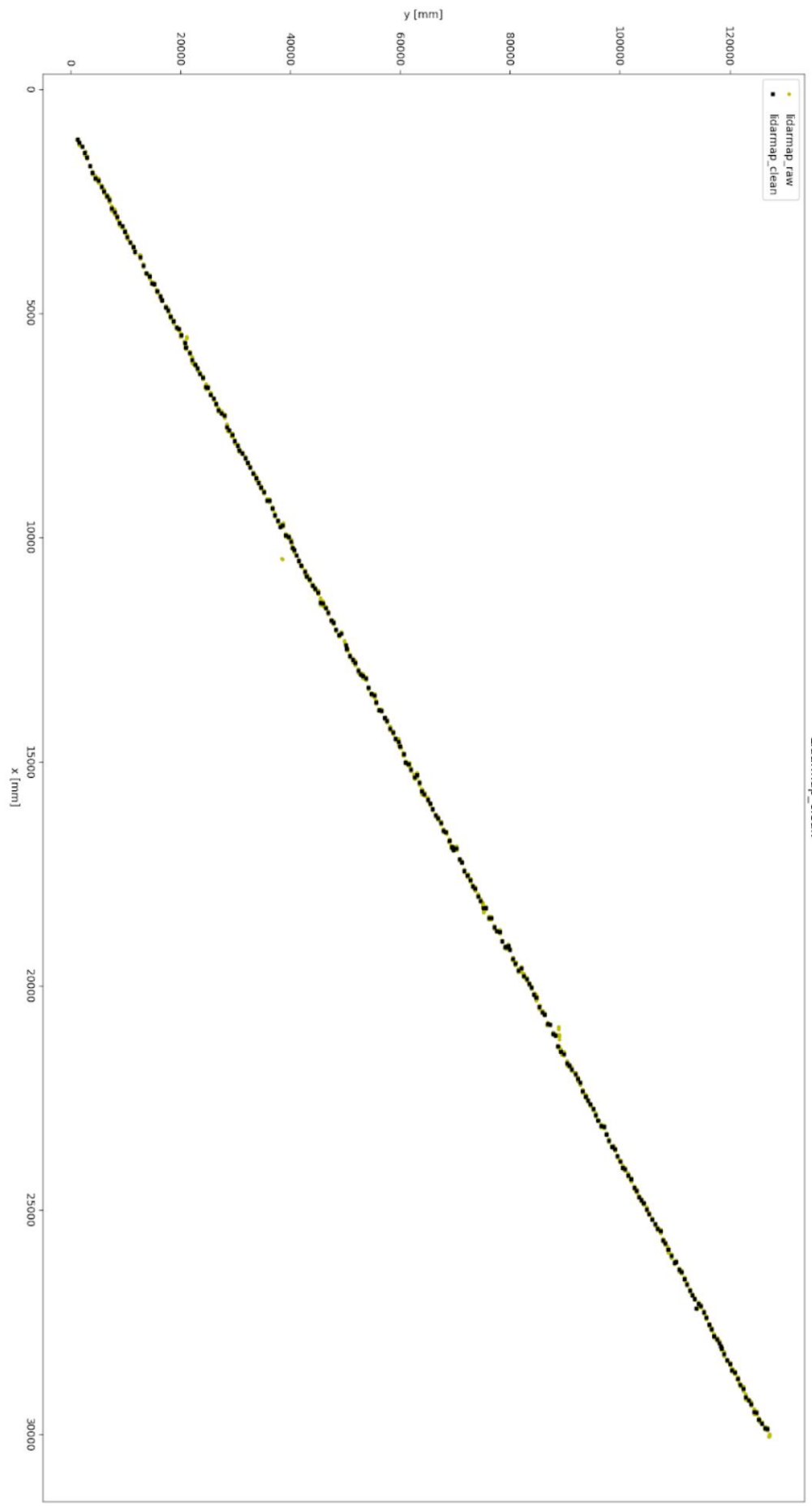
3 Apriorní mapa pro rozsáhlé měření

4 Vstupní data pro mapovou fúzi pro rozsáhlé měření

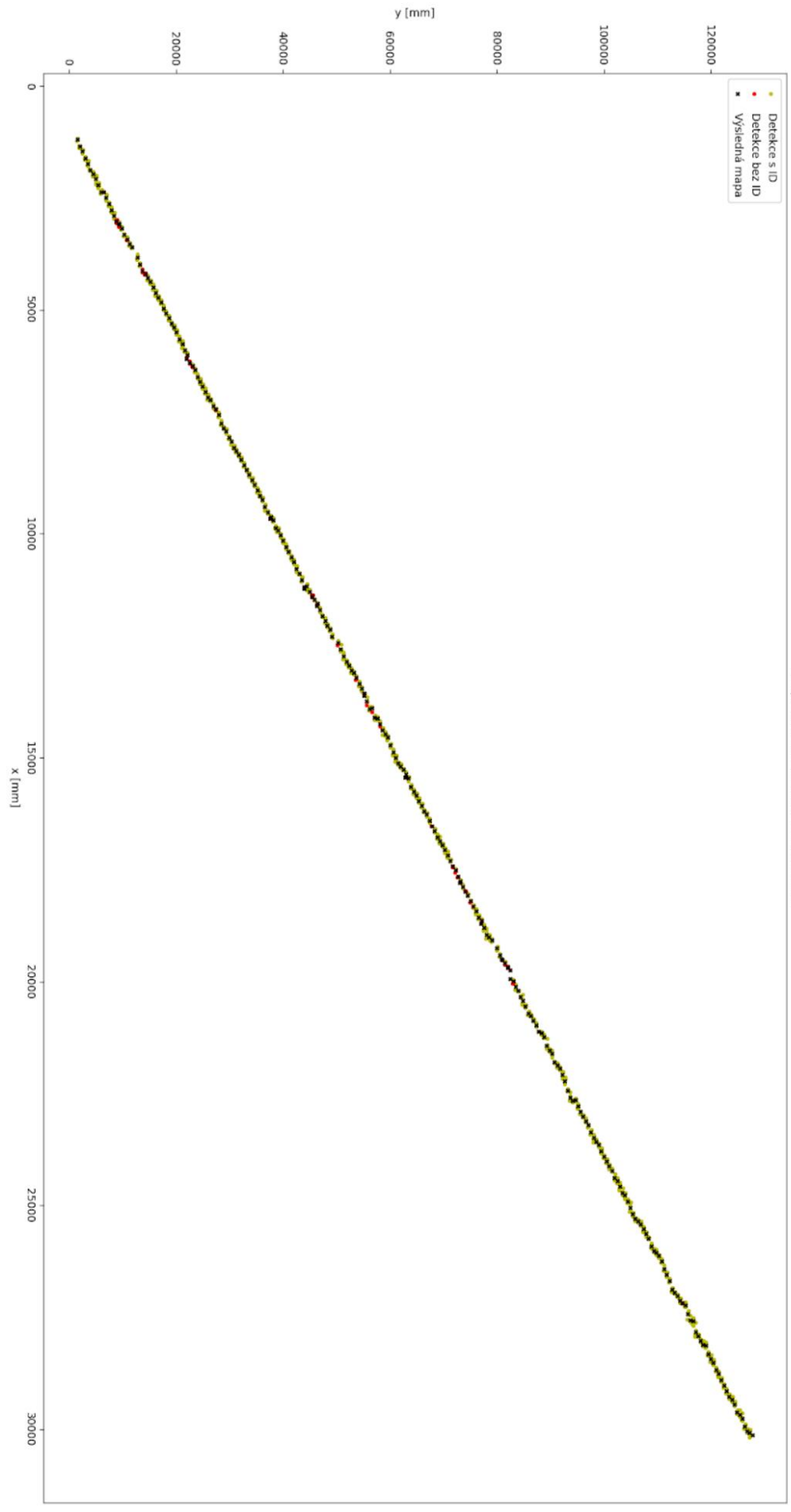
5 Výsledná mapa sadu pro rozsáhlé měření

6 Python program

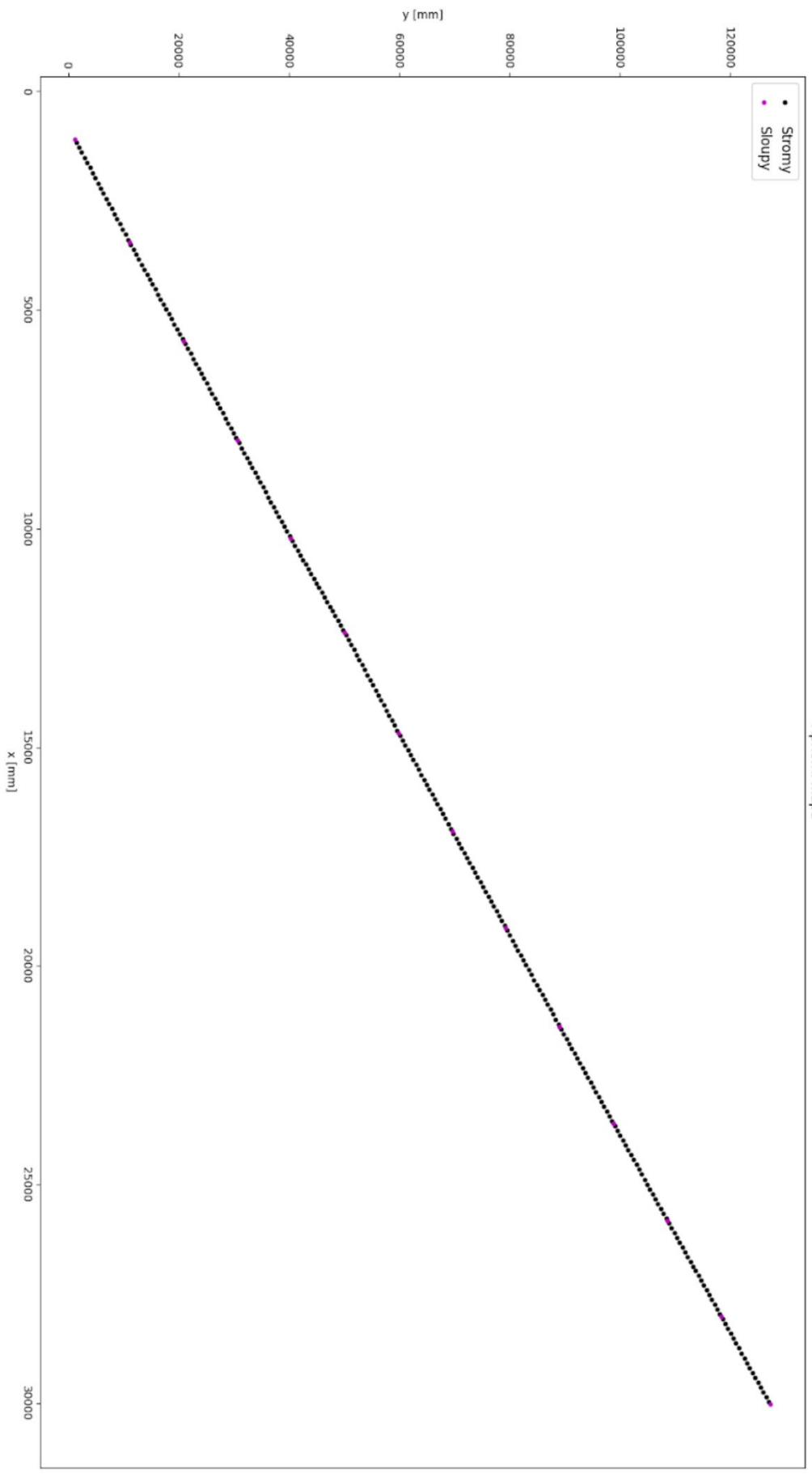
7 Video ze simulace HMI



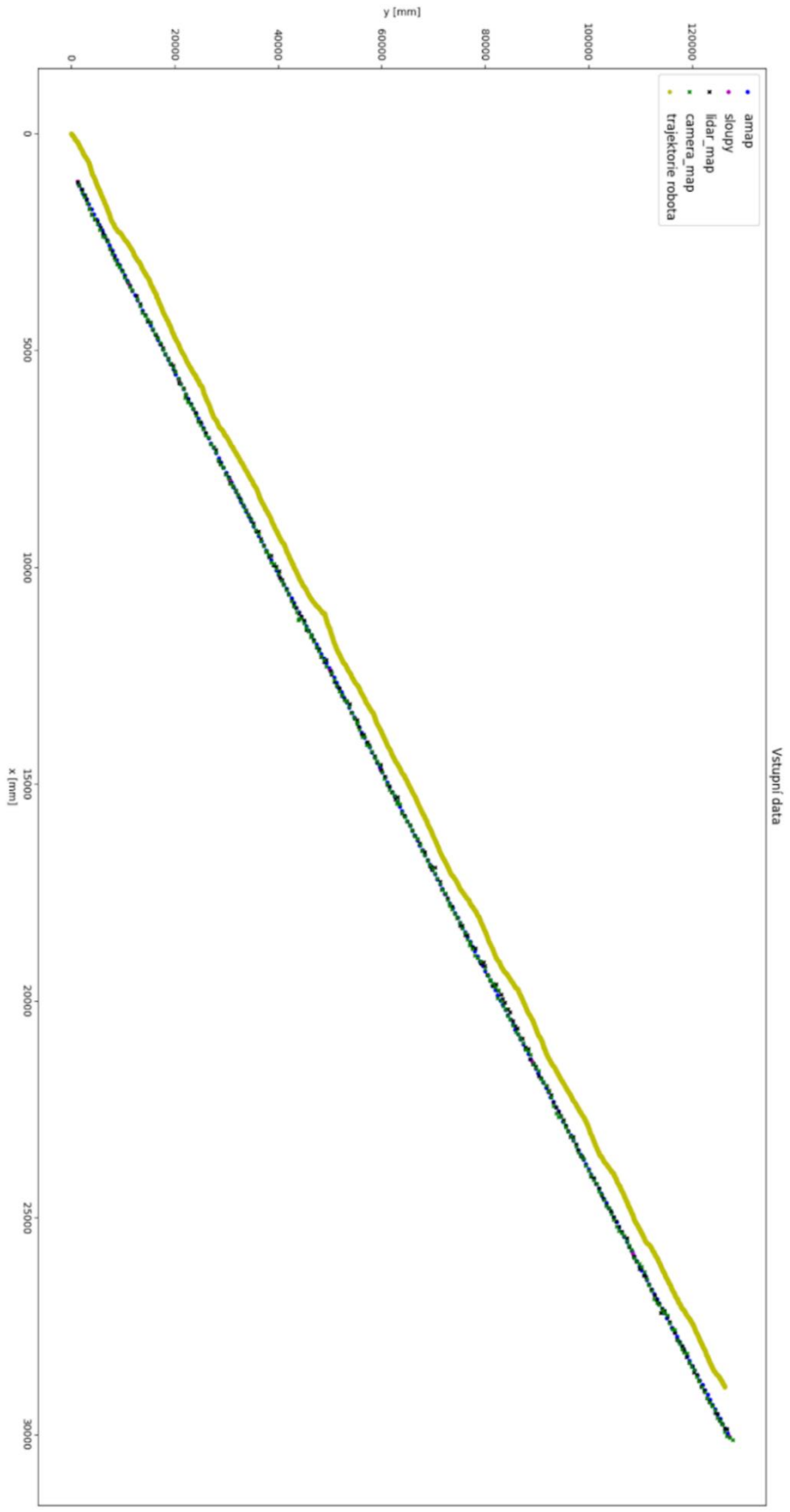
Příloha 1: Mapa z lidarových dat pro rozsáhlé měření



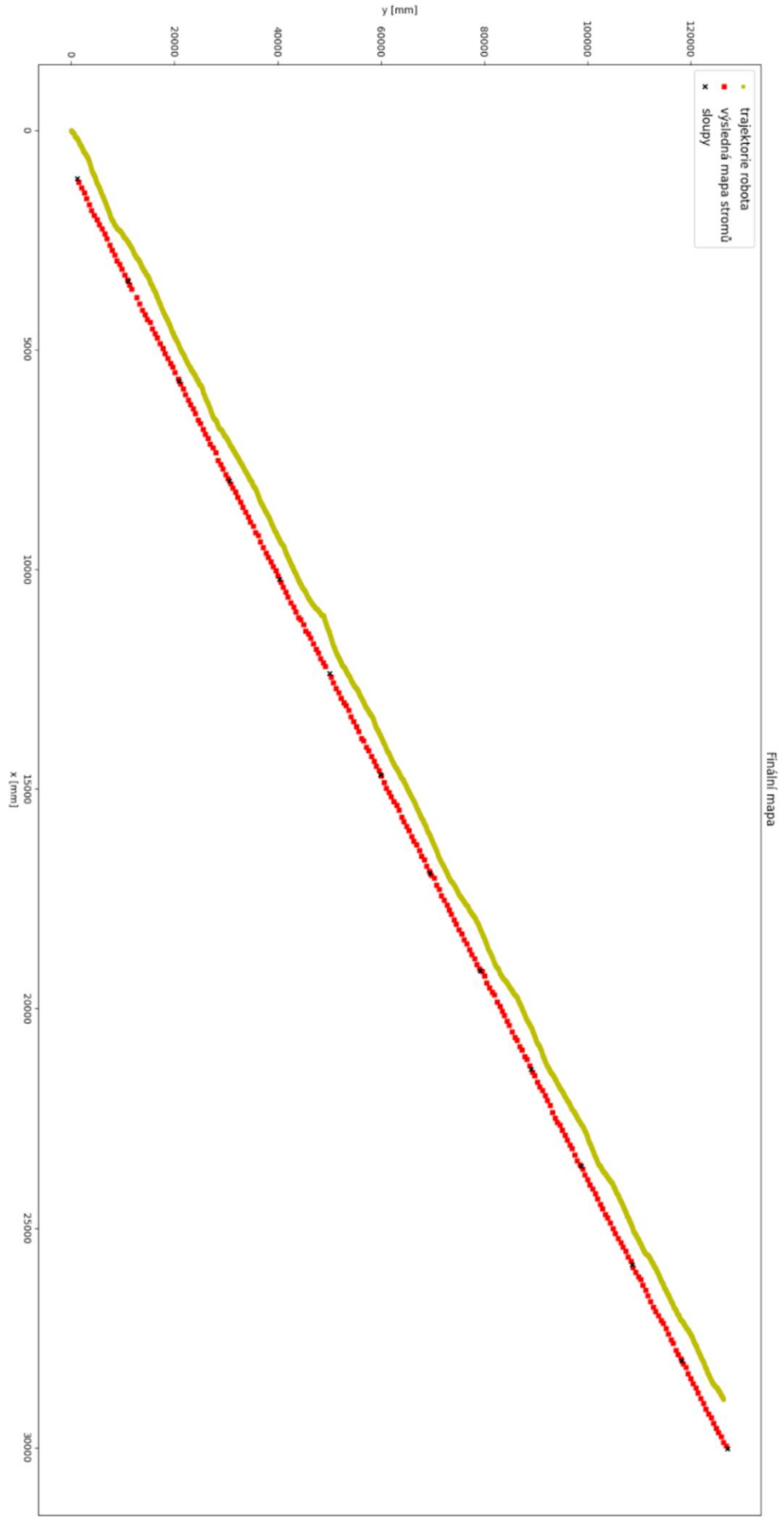
Příloha 2: Mapa z kamerového záznamu pro rozsáhlé měření



Příloha 3: Apriorní mapa pro rozsáhlé měření



Příloha 4: Vstupní data pro mapovou fúzi pro rozsáhlé měření



Příloha 5: Výsledná mapa sadu pro rozsáhlé měření