České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

Lokalizační systém pro mobilní roboty

Localization System for Mobile Robots

BAKALÁŘSKÁ PRÁCE

Vypracoval:	Tomáš Hromada
Studijní program:	Otevřená informatika
Specializace:	Základy umělé inteligence a počítačových věd
Vedoucí práce:	Ing. Jan Chudoba

Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hromada** Jméno: **Tomáš** Osobní číslo: **483629**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Otevřená informatika**

Specializace: **Základy umělé inteligence a počítačových věd**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Lokalizační systém pro mobilní roboty

Název bakalářské práce anglicky:

Localization System for Mobile Robots

Pokyny pro vypracování:

Cílem práce je návrh a implementace systému pro vizuální lokalizaci mobilních (pozemních či létajících) robotů v omezeném testovacím prostoru v reálném čase. Systém by měl kombinovat informaci z více kamer umístěných kolem daného prostoru. Roboty budou detekovány vizuálními značkami na nich umístěnými.
1. Nastudujte metody vizuální lokalizace a geometrické transformace projekce kamerou.
2. Proveďte rešerši souvisejících implemetovaných metod a jejich vlastností.
3. Navrhněte a implementujte lokalizační systém a proveďte vyhodnocení jeho přesnosti, robustnosti a dalších důležitých parametrů.

Seznam doporučené literatury:

[1] Šonka, Milan a Václav Hlaváč. Počítačové vidění. Praha: Grada, 1992.
[2] Edwin Olson. AprilTag: A robust and flexible visual fiducial system, in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), p. 3400-3407, IEEE, 2011.
[3] Krajník T., Nitsche M., Faigl, Vaněk, Saska, Přeučil, Duckett, Mejail. A practical multirobot localization system, Journal of Intelligent and Robotic Systems, Heidelberg, Springer (2014).
[4] Antonio Badal Regas, 3D Pose Estimation of Visual Markers, A Degree Thesis Submitted to ETSETB (UPC), TU Wien, 2015.
[5] Kolečkář David, Localization System for a Multi-robot System, bachelor thesis, Czech Technical University in Prague, Faculty of electrotechnical engineering, 2017.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Chudoba, inteligentní a mobilní robotika CIIRC

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

Datum zadání bakalářské práce: **05.01.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

Ing. Jan Chudoba	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. _____
Datum převzetí zadání

_____
Podpis studenta

Prohlášení

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

V Praze dne .....

.....
Tomáš Hromada

Poděkování

Rád bych poděkoval Ing. Janu Chudobovi za odborné vedení, za pomoc a rady při zpracování této práce.

Tomáš Hromada

Název práce:
Lokalizační systém pro mobilní roboty

Autor: Tomáš Hromada

Studijní program: Otevřená informatika
Specializace: Základy umělé inteligence a počítačových věd
Druh práce: Bakalářská práce

Vedoucí práce: Ing. Jan Chudoba
inteligentní a mobilní robotika CIIRC

Abstrakt: Tato bakalářská práce se zabývá návrhem a implementací vizuálního lokalizačního systému pro mobilní roboty. Navrhovaný systém využívá jednu nebo více statických kamer k odhadu polohy a rotace vizuálních značek umístěných na robotech. Jsou použity vizuální značky typu AprilTag, které se běžně využívají v robotice. Přidáním více kamer lze docílit větší přesnosti odhadu a zvětšení oblasti, kterou je systém schopen sledovat. Odhad polohy a rotace pomocí více kamer je realizován iterační metodou. Systém je schopen pracovat v reálném čase a sdílet vypočítané hodnoty dalším aplikacím. V práci jsou také popsány experimenty, na základě kterých byla určena přesnost a časová náročnost implementovaného systému.

Klíčová slova: vizuální lokalizační systém, soustava kamer, AprilTags

Title:
Localization System for Mobile Robots

Author: Tomáš Hromada

Abstract: This bachelor thesis deals with designing and implementation of a visual localization system for mobile robots. The proposed system uses one or more static cameras to estimate the position and rotation of visual markers placed on robots. Commonly used AprilTag visual markers are implemented to detect robots. By adding more cameras, greater estimation accuracy and increasing working area can be achieved. Estimation of position and rotation using multiple cameras is based on an iterative method. The system is able to work in real-time and share the calculated values with other applications. The thesis also contains the description of experiments, based on which the accuracy and runtime performance of the implemented system was determined.

Key words: visual localization system, multi-cameras system, AprilTags

Obsah

Seznam použitých zkratek

Úvod **1**

1 Příbuzné projekty **3**
1.1 WhyCon . . . 3
1.1.1 Přesnost . . . 4
1.2 Vizuální lokalizace pro experimentaci v mobilní robotice . . . 4
1.3 Vicon . . . 5
1.3.1 Přesnost . . . 5
1.3.2 Kalibrace . . . 5

2 Vizuální značky **7**
2.1 Výběr typu AR markeru pro lokalizační systém . . . 8
2.2 AprilTag . . . 8
2.2.1 Tag Rodiny . . . 9
2.2.2 Vhodná tag rodina . . . 9
2.2.3 Generování tagů . . . 10

3 Model kamery a projekce bodů **11**
3.1 Souřadnicové soustavy . . . 11
3.2 Model dírkové kamery . . . 12
3.2.1 Vnitřní parametry kamery . . . 12
3.2.2 Vnější parametry kamery . . . 13
3.2.3 Transformace souřadnicových soustav . . . 13
3.3 Model skutečné kamery . . . 14
3.3.1 Zkreslení obrazu čočkou . . . 14
3.3.2 Výpočet zkreslení . . . 14
3.4 Získání vnitřních parametrů kamery . . . 15

4 Odhad polohy a rotace pomocí jedné kamery **17**
4.1 PnP problém . . . 17
4.2 Výpočet polohy markeru ve světových souřadnicích . . . 18
4.3 Nedostatky detekce jednou kamerou . . . 19

5 Odhad polohy a rotace pomocí soustavy kamer **21**
5.1 Definice úlohy . . . 21
5.2 Popis algoritmu . . . 21
5.2.1 Přehled . . . 22
5.2.2 Levenberg–Marquardtova metoda . . . 22
5.3 Kalibrace multikamerového systému . . . 23

6 Implementace lokalizačního systému **25**

6.1 Volba použitých nástrojů . . . 25
6.1.1 Programovací jazyk . . . 25
6.1.2 AprilTag detektor . . . 25
6.1.3 OpenCV . . . 26
6.2 Funkce programu . . . 26
6.2.1 Odhad polohy markerů . . . 26
6.2.2 Kalibrace kamer . . . 26
6.2.3 Sdílení dat s jinými aplikacemi . . . 27
6.3 Struktura programu . . . 27
6.3.1 Systemové parametetry . . . 27
6.3.2 Detektor . . . 27
6.3.3 Odhad polohy a rotace . . . 28
6.3.4 Numerické výpočty . . . 28
6.3.5 TCP Server . . . 28
6.3.6 Main . . . 28
6.4 Vstupní argumenty a vstupní soubory . . . 29
6.4.1 Soubor s parametry kamery . . . 29
6.4.2 Soubor se seznamem kamer . . . 29
6.4.3 Soubor se seznamem markerů . . . 29

7 Experimentální ověření vlastností systému 31
7.1 Přesnost za použití 1 vs 2 vs 3 kamer . . . 31
7.1.1 Popis experimentu . . . 31
7.1.2 Výsledky . . . 32
7.2 Přesnost systému s fyzickými kamerami . . . 33
7.2.1 Popis experimentu . . . 33
7.2.2 Výsledky . . . 34
7.3 Časová náročnost implementace . . . 35
7.3.1 Popis experimentu . . . 36
7.3.2 Výsledky . . . 36
7.4 Velikost pokrytého prostoru . . . 37
7.4.1 Přesnost v závislosti na vzdálenosti . . . 37

Závěr 39

Bibliografie 41

Seznam obrázků 43

Seznam tabulek 44

Přílohy 45
A Obsah přiloženého CD . . . 45

Seznam použitých zkratek

DoF Stupně volnosti.

FoV Zorné pole.

FPS Snímky za sekundu.

LM Levenberg Marquardtova (iterační metoda).

Úvod

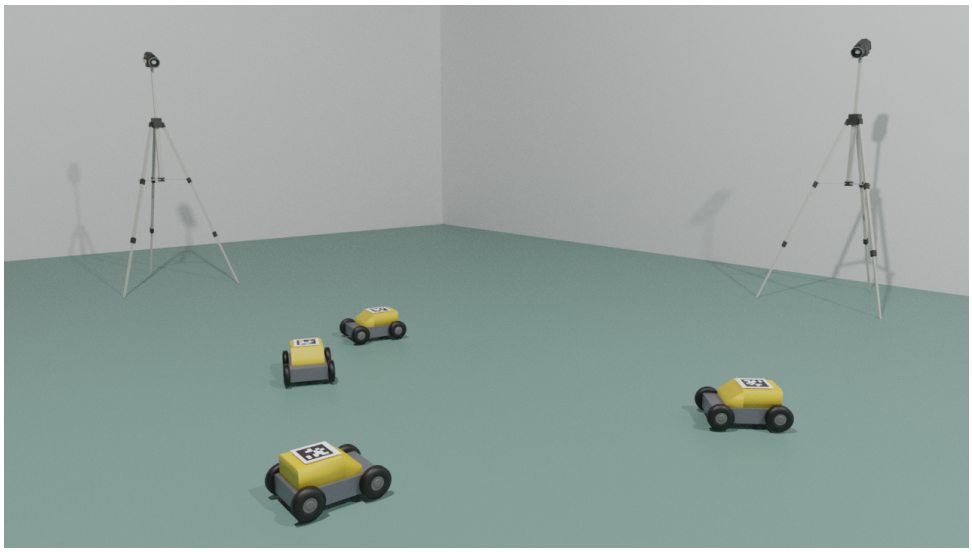
Má práce se zabývá návrhem a implementací vizuálního lokalizačního systému pro mobilní roboty, který by umožnil určení polohy a orientace robotů pomocí jedné nebo více statických kamer v daném omezeném prostoru.

Mezi hlavní problémy robotiky patří lokalizace robotů. Lokalizace je důležitá pro navigaci samotného robotu nebo pro jeho trénink či kontrolu pohybu. Existuje mnoho lokalizačních systémů, které se snaží tento problém řešit. Roboty mohou například využívat systém GPS, jehož přesnost může být pro některé účely příliš nízká. Jeho použití ve vnitřních prostorech je také problematické a to kvůli horšímu signálu. Další možností je například využít komerční motion-capture systém, např. Vicon, který může dosahovat přesnosti méně než 1 mm. Nevýhodou takového systému je jeho vysoká pořizovací cena.

Další využívanou možností je použití běžných kamer a papírových vizuálních značek (markerů), které jsou umístěny na robotech (obr. 1). Kamera je připojená k počítači, ve kterém se v obrázku z kamery detekují všechny markery, ze kterých je za podmínky, že známe parametry kamery a markerů, možné vypočítat jejich polohy a rotace. Problémem takového systému je jednak nižší přesnost a to nejvýrazněji ve směru pohledu kamery, je tedy obtížné odhadnout, jak daleko od kamery se marker nachází a za druhé také jeho nižší pokrytí scény. Kamery mají omezený zorný úhel, navíc marker může být nevhodně otočen takovým směrem, že ho nelze vidět kamerou. Tyto problémy se však dají vyřešit přidáním více kamer, které snímají scénu z různých úhlů (obr. 2). Právě takovýmto systémem se zabývám v této práci.

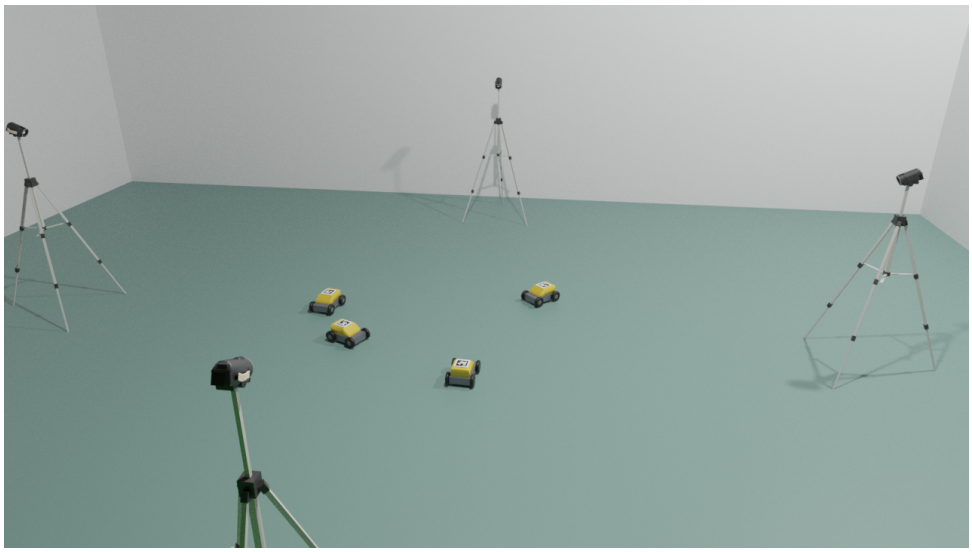
Cílem práce bylo navrhnout a posléze implementovat systém pro lokalizaci robotů v laboratorních prostorech pomocí běžných kamer. Systém musí být schopný detekovat pozemní roboty i létající roboty a spočítat jejich polohu a rotaci, musí tedy pracovat ve 3D. Systém by také měl pracovat v reálném čase. Jeho přesnost by se měla pohybovat v řádu jednotek centimetrů. Cílem práce je také stanovit pomocí experimentů vlastnosti navrhovaného systému.

První kapitola obsahuje stručný popis podobných existujících lokalizačních systémů. Druhá kapitola je věnována vizuálním značkám. Popisuji v ní vlastnosti značek AprilTag, které jsem vybral pro tento systém. Ve třetí kapitole se zaměřuji na matematický model kamery a projekci bodů. V následující čtvrté kapitole uvádím PnP problém a popisuji odhad polohy a rotace pomocí jedné kamery. V páté kapitole představuji navrhovaný algoritmus pro odhad polohy a rotace markerů s více kamerami. V šesté kapitole popisuji samotnou implementaci systému a použité nástroje. V sedmé kapitole uvádím popis a výsledky experimentů, pomocí kterých jsem testoval vlastnosti implementovaného systému.



Obrázek 1: Roboty s vizuálními značkami

,



Obrázek 2: Soustava kamer snímající roboty

,

Kapitola 1

Příbuzné projekty

Následuje krátký popis několika existujících lokalizačních systému pro mobilní roboty. Je zde zmíněna přesnost jednotlivých systémů, které slouží k porovnání s navrhovaným systémem.

1.1 WhyCon

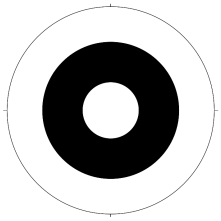
WhyCon [1] (obr. 1.1) je multirobotový lokalizační systém pro mobilní roboty založen na určování polohy robotů pomocí vizuální značek. Tento systém je vyvíjen ve spolupráci 3 univerzit: University of Buenos Aires, ČVUT a University of Lincoln, UK. Systém je určen pro rychlou detekci mnoha robotů v omezeném prostoru. K snímání scény je možné použít obyčejnou webkameru.

Obrázek 1.1: Lokalizační systém WhyCon
Zdroj: A Practical Multirobot Localization System [**krajnik**]

Autoři navrhli vlastní typ značek, jejichž detekce je rychlá a spolehlivá i při zhoršených světelných podmínkách. Tyto černo-bílé značky (obr. 1.2) se skládají ze 3 soustředných kruhů. Jednotlivé značky se mohou od sebe odlišovat různým poměrem poloměrů kruhů, a tím umožňují identifikovat jednotlivé roboty. Podle autorů to však může vést k menší přesnosti, protože algoritmus k počítání polohy tento poměr využivá. Tudíž pokud nezná přesný poměr poloměrů kruhů snižuje se přesnost. Značky lze vytisknout na obyčejný kancelářský papír pomocí běžné tiskárny.

Ze značek WhyCon nelze spočítat natočení, protože značky jsou středově souměrné. Tento problém řeší značky WhyCode [2] vycházející z WhyCon, ze kterých lze určit všechny 3 složky rotace.

Systém pracuje zatím pouze s jednou kamerou, ale autoři uvažují v budoucnu přidat podporu více kamer.



Obrázek 1.2: Kruhový marker WhyCon
Zdroj: A Practical Multirobot Localization System [1]

1.1.1 Přesnost

Autoři provedli několik testů pro ověření přesnosti systému. Vyzkoušeli několik kamer s různým rozlišením. Pro srovnání udávám výsledky testu za použití kamery s rozlišením 1280×720, protože takový typ kamer plánuji využít v navrhovaném lokalizačním systému. Velikost chyby systému WhyCon v 3D režimu dosahovala v průměru 3,7 cm při pohledu ze shora a při pohledu ze strany naměřili průměrnou chybu 5,7 cm. V testu se vzdálenost markerů od kamery průměrně pohybovala kolem 6 metrů. Výsledky jsou převzaty z práce [3].

1.2 Vizuální lokalizace pro experimentaci v mobilní robotice

V této bakalářské práci [4] od Tomáše Pivoňky je představen vizuální systém pro mobilní roboty postaven na systému WhyCon [3]. Systém využívá jednu kameru a detekuje kruhové značky WhyCon. Tento systém pracuje ve 3D a dokáže určit polohu i rotaci značek (6 DoF). V práci jsou popsány 2 metody na určení polohy. První z nich je postavena na výpočtu polohy ze tří bodů ležících na přímce. Druhá pak odhaduje polohu vyřešením P3P problému[1].

V této práci jsou také popsány experimenty k ověření přesnosti a jejich výsledky. Přesnost systému byla testována v aréně o rozměrech 3×3 m^2. Byla použita IP kamera s rozlišením 2592×1944 px. Průměr použitých kruhových WhyCon značek dosahoval 6 cm. Vzdálenost kamery od roviny, ve které se nacházely markery, byla 2,6 m. Přesnější z obou metod se ukázala být metoda řešící P3P problém. Chyba určení každé složky rotace byla u této metody menší než 1,5° (eulerovy úhly). Průměrná chyba určení polohy byla pod 1,5 cm a nepřesáhla 3 cm. Největší průměrná chyba (1,297 cm) byla naměřena v ose z, tedy v určování vzdálenosti kamery od značky. V osách x a y to bylo výrazně méně (0,351 cm a 0,204 cm).

Systém pracoval rychlostí 11FPS na PC s procesorem Intel® Core™ i7- 2630QM [4].

[1] PnP problém je popsán v sekci 4.1

1.3 Vicon

Vicon [5] je jedním z komerčních motion-capture lokalizačních systémů využívajících systém kamer k přesnému určení polohy a rotace markerů. Využívá se např. pro animaci 3D postav. Má využití také v průmyslu, v medicíně a v profesionálním sportu [6] ke sběru biomechanických dat. Systém Vicon využívá infračervené kamery, které snímají speciální marker z reflexivního materiálu. Systém se podobá navrhovanému systému v tom, že sledovaný objekt je snímán soustavou kamer. Rozdíl je, že Vicon (i ostatní systémy) pracují v infračerveném světle a využívají speciální markery. Tyto kamery také dosahují mnohem vyšší frekvence snímání a to 160 FPS (kamera Vicon MX-40+) na rozdíl od běžných kamer, které snímají 30 nebo 60 FPS.

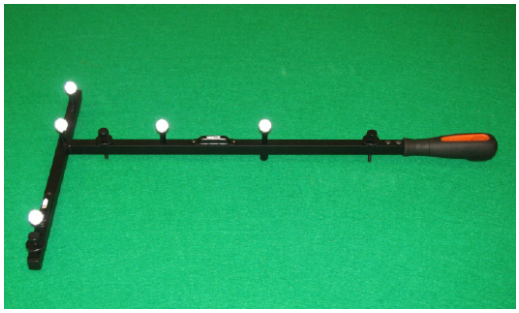
1.3.1 Přesnost

Vicon i ostatní motion-capture systémy dosahují vysoké přesnosti. Konkrétní přesnost záleží na počtu kamer a dalších parametrech. Systém Vicon dosahuje přesnosti menší než 1 mm. V této práci [7] byla testována přesnost systému, pro statické objekty průměrná absolutní chyba dosahovala 0,15 mm. Velikost testovací oblasti byla $2 \times 1,5 \times 1\ m^3$ a bylo použito 8 kamer.

1.3.2 Kalibrace

Zajímavým pro tento projekt je způsob kalibrace, který Vicon využívá. Systém Vicon stejně jako navrhovaný systém používá soustavu kamer. Před spuštěním systému je potřeba určit polohy a rotace všech kamer. Získat tyto data může být komplikované.

Systém Vicon určí polohu a rotaci kamer automaticky zpracováním vzorku dat. Data vznikají záznamem pohybů kalibrační hůlky (obr. 1.3) v prostoru. Systém detekuje markery na hůlce několika kamerami a na základě toho odhaduje pózy kamer. Kalibrace kamer se pohybem hůlky před kamerami postupně zpřesňuje.



Obrázek 1.3: Vicon kalibrační hůlka s 5 markery
Zdroj: Utrecht Multi-Person Motion (UMPM) benchmark [8]

Kapitola 2

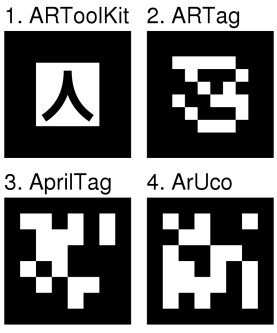
Vizuální značky

V této kapitole popíši vizuální značky, které používá lokalizační systém k detekci robotů v obrázcích. Zmíním různé typy značek, vysvětlím, proč jsem si zvolil značky typu AprilTag a popíši jejich vlastnosti.

AR[1] markery jsou speciální vizuální značky designované ke snadnému rozpoznání ve snímcích pořízených běžnou kamerou. Tyto markery většinou využívají černé a bílé segmenty k zakódování identifikátoru markeru a jeho orientaci. Markery je možné vytisknout na běžné tiskárně na obyčejný papír.

AR Markery byly vyvinuty tak, aby pomocí nich bylo možné určit polohu a rotaci kamery (6 DoF), se kterou byl snímek s markerem pořízen. Po zjištění pózy kamery lze například do obrázku vykreslit model tak, aby se jevil jako součást původní scény. Dalším využitím je nalézt polohu markeru vůči kameře invertováním pózy kamery, což je možné využít v lokalizačních systémech [9]. Sledované objekty mohou mít na sobě umístěné markery s různými identifikátory, což umožňuje rozpoznávání jednotlivých objektů.

Existuje množství druhů markeru. Jako příklad uvedu tyto: ARTag [10], ARToolkit [11], ArUco [12] a AprilTag [13]. Na obrázku 5.1 můžeme vidět ukázky jednotlivých markerů. Markery nemusí být jen čtvercové, příkladem kruhového markeru je značka (obr. 1.2) používaná v systém WhyCon popsaném v kapitole 1.1.



Obrázek 2.1: Porovnání markerů různých typů
Zdroj: CMG Lee [14]

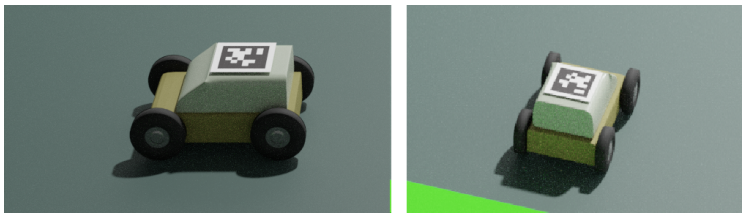
Využití AR markerů je široké. Pomocí AR markerů byl vyvinut např. systém na sledování pohybů operačního nože [15], který dosahoval na vzdálenost 70 cm maximální chyby 9,3 mm. AR markery se také široce využívají v lokalizačních a navigačních systémech pro roboty. Například byl vytvořen systém pro automatické

[1] Augmented reality = rozšířená realita

přistávání a navigaci dronů za pomoci ArUco markeru [16] nebo systém využívající AprilTag markery pro navigaci podvodních plavidel [17].

2.1 Výběr typu AR markeru pro lokalizační systém

V navrhovaném lokalizačním systému jsem se pro lokalizaci robotů rozhodl využít AR markery, které budou umístěny viditelně na robotech (obr. 2.2). Při výběru konkrétního typu jsem se řídil následujícími požadavky. Detektor markerů musí být robustní, měl by být schopen detekovat marker ve větší vzdálenosti (více než 2 metry), při horších světelných podmínkách i při pohledu z ostrých úhlů, je totiž žádoucí, aby byl marker detekován co možná nejvíce kamerami, které sledují scénu z různých úhlů. Zároveň byl měl příslušný detektor pracovat dostatečně rychle, protože lokalizační systém je navrhován pro práci s více kamerami v reálném čase, tudíž detektor zpracovává v každé iteraci běhu systému snímek z každé kamery.



Obrázek 2.2: Model robota s umístěným AprilTag markerem

,

Jako dva nejvhodnější typy jsem určil ArUco a AprilTag. Oba tyto typy markerů byly použity v několika podobných projektech. ArUco využily například tyto projekty [9], [16]. Naopak AprilTag využili autoři systému pro lokalizaci multikoptér [18]. Mezi další projekty využívající AprilTag markery patří tento [19] nebo již zmíněný navigační systém pro podmořská plavidla [17].

Práce [20] porovnávala vlastnosti markerů ve zhoršených podmínkách. V práci autoři dospěli k závěru, že nejlepší volbou se jeví AprilTag, který se ukázal být velmi robustní. AprilTag byl detekován na větší vzdálenost a vyžadoval menší velikost než ArUco. V této práci ale také došli k závěru, že ArUco je výrazně rychlejší než AprilTag.

Otestoval jsem proto rychlost detekce na svém notebooku (Intel i5-8250U 1,6 GHz, 8 GB RAM). Zpracovat snímek o velikosti 1920×1080 pixelů trvalo pruměrně 96 ms a snímek o velikosti 1280×720 58 ms. Tato rychlost se mi zdála dostačující, proto jsem se rozhodl pro AprilTag markery, které vykazují vetší robustnost než ArUco i za cenu možné nižší rychlosti. Případné problémy s výkonem lze řešit například více-vláknovým programováním.

2.2 AprilTag

AprilTag je systém vizuálních značek vyvinutý na Michiganské univerzitě. Jeho výhodou oproti starším systémům je větší robustnost při částečném zakrytí, různém nasvícení nebo pohledech z ostrých úhlů. Tento systém je také dostatečně rychlý, proto aby ho bylo možné využít pro real-time aplikaci [13].

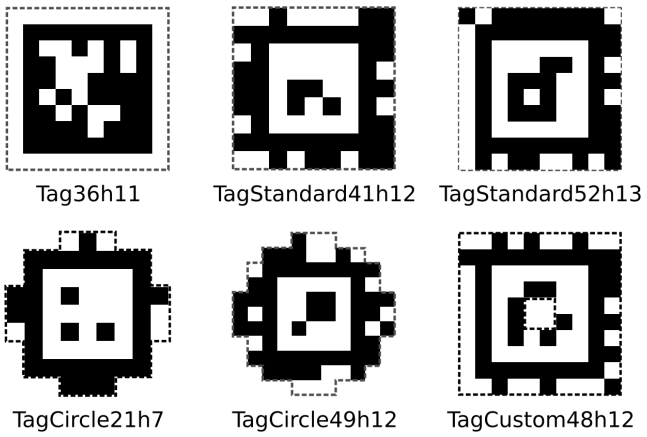
AprilTag systém je běžně využíván v robotice, což dokazují výše zmíněné projekty. Existuje několik implementací detektoru v různých jazycích (C, Python, Matlab a Julia) [21]. Jedná se o open-source projekt vydaný pod BSD licencí.

2.2.1 Tag Rodiny

Existuje mnoho rodin april tagů[2], rodiny se navzájem liší počtem data modulů, tvarem nebo hammingovou vzdálenosti dvou tagů ze stejné rodiny. April Tagy mohou být i kruhové (TagCircle21h7, TagCircle49h12 na obrázku 2.3) nebo mohou obsahovat otvor uvnitř tagu (TagCustom48h12 na obrázku 2.3).

V názvu rodiny první číslo reprezentuje počet data modulů (bitů), druhé pak značí minimální hammingovou vzdálenost mezi dvěma tagy. Například Tag36h11 (obr. 2.3) obsahuje 6×6 data bodů (tedy 36) a hammingova vzdálenost je 11.

Různé rodiny tagů mohou být vhodné pro různé aplikace. Se zvětšující se minimální hammingovou vzdáleností klesá pravděpodobnost záměny tagu s jiným, na druhou stranu ale klesá počet unikátních tagů v jedné rodině, které lze vygenerovat. V tabulce 2.1 je zmíněno několik rodin s počty unikátních tagů. Výhodou rodin s menším počtem datových modulů je menší rozlišení výsledného markeru, což umožňuje marker detekovat z větší vzdálenosti [22]. Nevýhodou je, že mohou častěji vznikat chybně pozitivní detekce tagů v obrázku.



Obrázek 2.3: Různé Tag Family April Tagů
Zdroj: april.eecs.umich.edu/software/apriltag

2.2.2 Vhodná tag rodina

V AprilTag dokumentaci [23] je popsána základní heuristika pro vybrání ideální tag rodiny pro konkrétní účely projektu. Pro standardní účely by měla postačit rodina *TagStandard41h12*. Pokud je potřeba větší množství unikátních tagů, je doporučována rodina *TagStandard52h13* (48714 unikátních tagů). Jestli máme naopak potřebu maximalizovat využitý prostor na malých kruhových objektech, vybereme *TagCircle49h12* nebo *TagCircle21h7*. Pokud potřebujeme, aby tagy byly kompatibilní s ArUco tagy, měli bychom vybrat *Tag36h11*.

[2] Anglicky Tag family

Tabulka 2.1: Počet tagů pro různé rodiny

Zdroj: https://github.com/AprilRobotics/apriltag-imgs

TagFamily	Počet data bitů	Min. Hamming	Počet tagů
16h5	16	5	30
25h9	25	9	35
Circle21h7	21	7	38
Circle49h12	49	12	65698
36h11	36	11	587
41h12	41	12	2115
Standard52h13	52	13	48714

2.2.3 Generování tagů

Několik April tagových rodin je už předgenerováno a lze je stáhnout z oficiálního githubu AprilRobotics/apriltag-imgs. Také je k dispozici program napsaný v jazyce Java sloužící pro generování tagů. Tento program je k dispozici na githubu AprilRobotics/apriltag-generation. Vygenerovaný tag je pak možné zvětšit do potřebné velikosti a vytisknout na kancelářský papír či čtvrtku.

Kapitola 3

Model kamery a projekce bodů

V této kapitole popíši model kamery, se kterým je možné vypočítat projekce bodu danou kamerou. Toto využijeme při odhadu polohy markeru v následujících kapitolách.

Abychom mohli spočítat z detekovaných rohů markeru jeho polohu a rotaci, musíme nejdříve zavést metodu, která je schopná 3D bod ve světových souřadnicích zprojektovat do 2D plochy obrázku. K výpočtu této projekce je potřeba si popsat model kamery. Nejdříve zavedeme model dírkové kamery, který poté rozšíříme o zkreslení, a tím získáme model skutečné kamery.

3.1 Souřadnicové soustavy

Při projekci světových bodů do obrázkových musíme souřadnice postupně převádět mezi 4 různými souřadnicovými soustavami. Popis těchto souřadnicových soustav vychází z této práce [24], kde jsou podle mého názoru tyto souřadnicové systémy dobře popsány, a také z oficiální opencv dokumentace [25].

1. **Světové souřadnice**

   Světové souřadnice jsou 3 rozměrné. Počátek této soustavy záleží na našem uvážení, může to být například střed místnosti nebo střed jedné z kamer. Je zvykem, že jedna jednotka odpovídá jednomu metru. V této práci tyto souřadnice značíme: (X_W, Y_W, Z_W) a používáme jednotky metry.

2. **Kamerové souřadnice**

   Kamerové souřadnice jsou také 3 rozměrné. Mají počátek v optickém středu[1] kamery. Osa z je zarovnaná s optickou osou kamery, míří tedy dopředu, osa y odpovídá směru dolů z pohledu kamery. Jedna jednotka odpovídá stejné jednotce jako ve světovém souřadnicovém systému. Souřadnice budeme značit: (X_C, Y_C, Z_C)

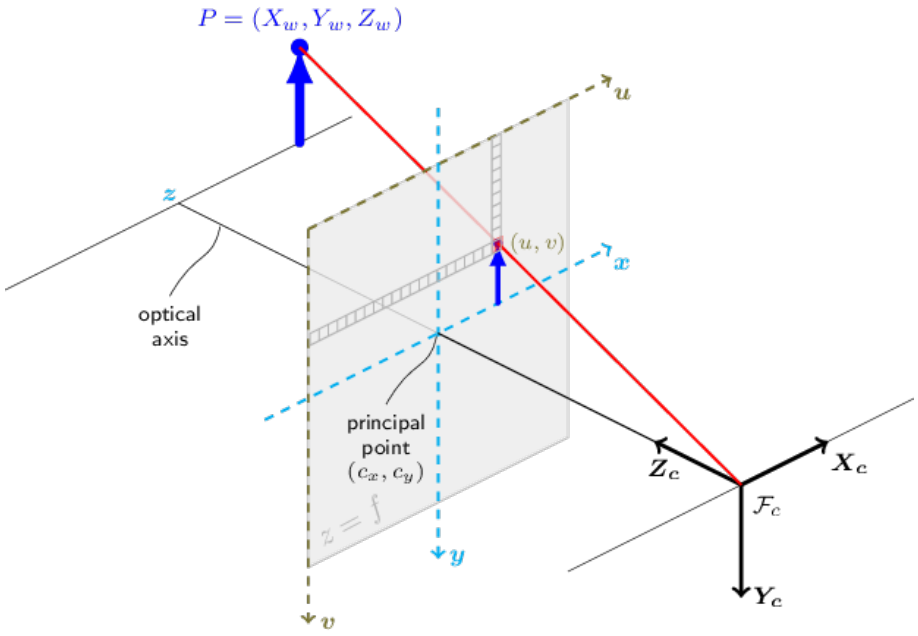
3. **Souřadnice obrázkové roviny**

   Tento souřadnicový systém je 2 rozměrný, střed odpovídá středu kamerových souřadnic a osy x, y také odpovídají kamerovým osám. Značíme: (x', y').

4. **Obrázkové souřadnice**

[1] Anglicky Principle point

Obrázkové souřadnice jsou 2 rozměrné, odpovídají jednotlivovým pixelům v obrázku. Počátek se nachází v levém horním rohu obrázku (někdy se také umisťuje do levého dolního rohu). Souřadnice jsou udávány v pixelech a značíme je (u, v).



Obrázek 3.1: Model dírkové kamery s popisem souřadnicových systémů
Zdroj: docs.opencv.org/3.4/d9/d0c/group$_{c}alib3d.html$

3.2 Model dírkové kamery

Model dírkové kamery[2] (obr. 3.1) popisuje projekci 3D světových bodů do 2D obrazové roviny bez zkreslení. Převod 2D bodu na 3D lze napsat pomocí maticových operací [25].

Pokud chceme zprojektovat bod kamerou, potřebujeme znát parametry konkrétní kamery, tyto parametry můžeme rozdělit na vnitřní a na vnější.

3.2.1 Vnitřní parametry kamery

Mezi vnitřní[3] parametry kamery patří ohnisková vzdálenost f a optický střed c. Ohnisková vzdálenost je definována jako vzdálenost ohniska od optického středu. Optický střed je bod na obrázkové rovině, do kterého se promítne střed perspektivy [26]. Jak získat tyto parametry u fyzické kamery je popsáno v této sekci 3.4.

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

[2] Anglicky pinhole camera model
[3] Anglicky intrinsic

kde: f = ohnisko kamery v pixelech
c = optický střed v pixelech
A = matice vnitřních parametrů kamery

3.2.2 Vnější parametry kamery

Vnější[4] parametry kamery popisují umístění kamery ve světovém prostoru, tedy rotaci a vektor posunutí kamery.

$$W = \begin{bmatrix} rot_{1_1} & rot_{1_2} & rot_{1_3} & t_x \\ rot_{2_1} & rot_{2_2} & rot_{2_3} & t_y \\ rot_{3_1} & rot_{3_2} & rot_{3_3} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

kde: rot = rotační matice kamery
t = vektor posunutí kamery
W = matice vnějších parametrů kamery

3.2.3 Transformace souřadnicových soustav

Nejdříve převedeme bod ve světových souřadnicích do homogenních souřadnic, tedy přidáme 1, vznikne tak 4 rozměrný vektor. Tento vektor zobrazíme pomocí matice W a získáme tak souřadnice v kamerových souřadnicích.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = W \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.1}$$

kde: X_W, Y_W, Z_W = světové souřadnice bodu
W = matice vnějších parametrů kamery
X_C, Y_C, Z_C = kamerové souřadnice bodu

Dále převedeme bod do souřadnic obrázkové roviny.

$$x' = X_C / Z_C$$
$$y' = Y_C / Z_C$$

Nakonec zobrazíme bod maticí vnitřních parametrů a získáme tak bod v obrázkových souřadnicích. Víme tedy, kam se bod promítne ve výsledném obrázku.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{3.2}$$

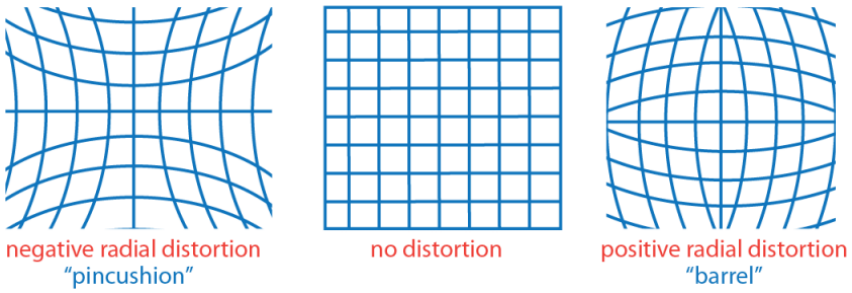
kde: u, v = obrázkové souřadnice projekce bodu v pixelech
A = matice vnitřních parametrů kamery

[4] Anglicky extrinsic

Takto jsme získali bod v obrázku, do kterého se promítne bod z reálného světa, pokud zanedbáme zkreslení způsobené čočkou. Abychom našli opravdový obraz bodu promítnutý skutečnou kamerou, musíme model rozšířit.

3.3 Model skutečné kamery

Model skutečné kamery vychází z modelu dírkové kamery. Liší se jen tím, že se před výpočtem obrázkových souřadnic aplikuje zkreslení[5].



Obrázek 3.2: Ukázka radiálního zkreslení
Zdroj: au.mathworks.com/help/vision/ref/cameraparameters-class.html

3.3.1 Zkreslení obrazu čočkou

Reálné čočky v kamerách obraz zkreslují. Rovné čáry se na obrazu mohou zdát zakřivené, stejně jako můžeme vidět na tomto obrázku 3.2. Při projekci bodů je potřeba s tímto jevem počítat a tento jev eliminovat. Hlavní typ zkreslení, který ovlivňuje obrázek většinou nejznatelněji, je radiální a pak částečně také tangenciální zkreslení [25].

Tato zkreslení můžeme charakterizovat 5 parametry k_1, k_2, p_1, p_2, k_3. Koeficienty k_1, k_2, k_3 reprezentují radiální zkreslení a parametry p_1, p_2 tangenciální. Parametry radiálního zkreslení lze popsat přesněji pomocí více parametrů, také existují další typy zkreslení (např. hranové), ale pro účely tohoto lokalizačního systému postačí 5 parametrů výše uvedených.

3.3.2 Výpočet zkreslení

Pro výpočet radiálního a tangenciálního zkreslení použijeme následující vzorec. Další typy zkreslení v tomto projektu zanedbáváme.

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2) \\ y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y' \end{bmatrix} \tag{3.3}$$

kde:
r $= \sqrt{x'^2 + y'^2}$
x', y' = souřadnice v obrázové rovině
k_1, k_2, k_3 = koeficienty raidálního zkreslení
p_1, p_2 = koeficienty tangenciální zkreslení
x'', y'' = souřadnice po aplikaci zkreslení

[5] Anglicky distortion

Dále převedeme body ze souřadnic obrazové roviny do obrázkových souřadnic tak jako u dírkové kamery.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} \tag{3.4}$$

kde: x'', y'' = souřadnice po aplikaci zkreslení
A = matice vnitřních parametrů kamery
u, v = obrázkové souřadnice projekce bodu v pixelech

Takto můžeme získat projekci bodu skutečnou kamerou, která by měla odpovídat realitě dostatečně pro účely tohoto projektu. Tato metoda je implementována v knihovně opencv pod názvem cv::projectPoints().

3.4 Získání vnitřních parametrů kamery

Zjištění vnitřních parametrů kamery může být obtížné a to zejména zjištění koeficientů zkreslení. Některé parametry můžeme spočítat, vypočítat lze např. ohniskovou vzdálenost v pixelech, pokud známe ohniskovou vzdálenost v jednotkách délky a velikost senzoru, pomocí následujících vzorců:

$$f_x = f \times width/s_x$$
$$f_y = f \times height/s_y$$

kde: f = ohnisková vzdálenost kamery v jednotkách délky
$width$ = šířka výsledného obrázku v pixelech
$height$ = výška výsledného obrázku v pixelech
s_x, s_y = šířka a výška senzoru ve stejných jednotkách jako f

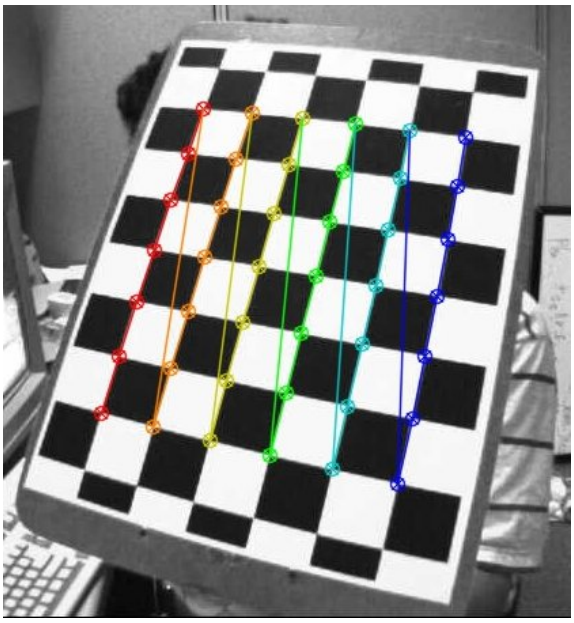
Dále se můžeme pokusit aproximovat optický střed jako střed obrázku., což ale nemusí odpovídat realitě, některé čočky mají optický střed posunutý mimo střed obrázku.

$$c_x \approx \frac{width}{2}$$
$$c_y \approx \frac{height}{2}$$

kde: $width$ = šířka výsledného obrázku v pixelech
$height$ = výška výsledného obrázku v pixelech
c_x, c_y = souřadnice optického středu

Pokud však potřebujeme zjistit parametry přesněji včetně koeficentů zkreslení, je vhodné použít metodu kalibrace, které jsou předloženy fotografie dobře definovaného kalibračního vzoru (např. šachovnice jako na obrázku 3.3) z různých úhlů a vzdáleností. Tato metoda spočítá všechny parametry kamery: ohniskové vzdálenosti, optický střed a koeficienty radiální a tangenciální zkreslení.

Tato metoda je implementována v knihovně opencv. [27] Vyžaduje alespoň 10 fotografií kalibračního vzoru.



Obrázek 3.3: Kalibrační šachovnice sloužící pro zjištění parametrů kamery
Zdroj: docs.opencv.org [27]

Kapitola 4

Odhad polohy a rotace pomocí jedné kamery

Základní úloha navrhovaného lokalizačního systému je odhadnout polohu markeru použitím jedné kamery. Algoritmus pro odhad polohy a rotace markeru z více kamer využívá tuto úlohu pro určení počátečních hodnot.

Systém na vstupu dostane snímek scény obsahující marker. Detektor v tomto snímku nalezne 4 body odpovídající rohům markeru. Z těchto čtyř bodů je možné po vyřešení PnP problému získat polohu a rotaci kamery vůči markeru. Přepočítáním souřadnic lze také nalézt polohu a rotaci markeru vůči kameře nebo vůči počátku souřadnicové osy.

4.1 PnP problém

Známe obrázkové 2D body, na kterých se nachází rohy detekovaného markeru. Tyto body nám vrátí detektor april tagů. Dále známe k těmto 2D bodům odpovídající 3D body ve světových souřadnicích. Tyto 3D body máme k dispozici, protože známe fyzické rozměry markeru, který si můžeme umístit do počátku a zarovnat ho se souřadnicovými osami. Potřebujeme nalézt parametry (poloha, rotace) kamery, tak aby kamera zadané 3D body zprojektovala na odpovídají 2D obrázkové body. Tato úloha se nazývá Perspective-n-Point (PnP) problém.

PnP problém (v našem případě n=4) můžeme napsat jako neomezenou minimalizační úlohu (4.1):

$$\min_{R_M, t_M} \quad \sum_{i=1}^{4} \left\| proj(R_M, t_M, X_i) - Y_i \right\|^2 \tag{4.1}$$

$$X = \left(\begin{bmatrix} -s \\ 0 \\ -s \end{bmatrix}, \begin{bmatrix} s \\ 0 \\ -s \end{bmatrix}, \begin{bmatrix} s \\ 0 \\ s \end{bmatrix}, \begin{bmatrix} -s \\ 0 \\ s \end{bmatrix} \right) \tag{4.2}$$

kde: R_M = rotační matice kamery
t_M = vektor posunutí kamery
X_i = souřadnice i-tého bodu ve světových souřadnicích
Y_i = souřadnice i-tého bodu v obrázku
$2s$ = fyzický rozměr markeru v metrech

K nalezení řešení se využívají iterační metody, např. RANSAC nebo Levenberg-Marquardtova metoda [28].

Funkce na řešení tohoto problému je implementována v knihovně opencv (solvePnP [28]).

4.2 Výpočet polohy markeru ve světových souřadnicích

Vyřešením PnP problému (použitím funkce solvePnP) zjístíme polohu kamery vůči markeru. Potřebujeme však znát polohu markeru ve světových souřadnicích, proto musí být vektor souřadnic vynásoben maticí zobrazení, které převede všechny rohy markeru z výchozí polohy 4.2 do světových souřadnic. Pro tyto účely nejdříve převedeme souřadnice rohů do homogenních souřadnic a vytvoříme tak matici P_0 (4.3) s rozměry 4×4.

$$P_0 = \begin{bmatrix} -s & s & s & -s \\ 0 & 0 & 0 & 0 \\ -s & -s & s & s \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.3}$$

kde: $2s$ = délka hrany markeru v metrech

Poté zavedeme následující zobrazení 4.4, které převede rohy markeru zadané v souřadnicích vzhledem k středu markeru do světových souřadnic.

$$M = \begin{bmatrix} R_C & t_C \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} R_M & t_M \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_W & t_W \\ 0 & 1 \end{bmatrix} \tag{4.4}$$

$$P_W = MP_0 \tag{4.5}$$

kde: R_C = rotační matice kamery
t_C = vektor posunutí kamery
R_M = rotační matice markeru vůči kameře
t_M = vektor posunutí markeru vůči kameře
R_W = rotační matice markeru ve světových souřadnicích
t_W = vektor posunutí markeru ve světových souřadnicích
M = zobrazí, které převede bod do světových souřadnic
P_0 = Matice s rohy markeru umístěném v počátku bez rotace
P_W = matice bodů markera ve světových souřadnicích

Tímto způsobem lze vypočítat polohu všech rohů markeru ve světových souřadnicích. Souřadnice rohů se nachází po sloupcích v matici P_W (4.5). Pokud chceme znát jen polohu markeru a jeho rotaci, najdeme ji v matici M (4.4) v posledním sloupci (t_W). V matici M najdeme také rotační matici (R_W). Rotační matici lze převést na rotační vektor, např. pomocí opencv funkce cv:Rodrigues.

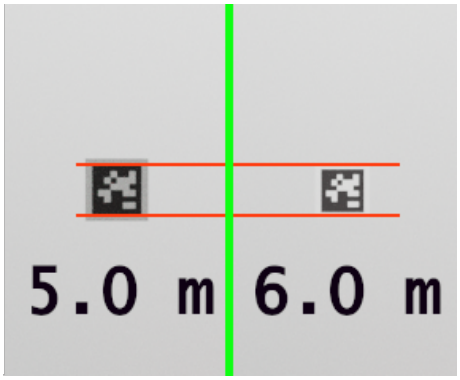
Tento vektor posunutí a rotační vektor můžeme považovat za výsledek. Takto jsme odhadli polohu a rotaci markeru systémem s jednou kamerou.

4.3 Nedostatky detekce jednou kamerou

Při odhadu polohy markeru systémem pouze s jednou kamerou narazíme na 2 problémy, které komplikují přesný odhad.

První z nich je ten, že se budeme potýkat s problémem správně určit vzdálenost markeru od kamery. Při projekci totiž ztrácíme jeden rozměr a to právě hloubku. Tento problém bude znatelnější, pokud se marker bude nacházet blízko optické osy kamery a pokud bude marker orientován čelem ke kameře.

Pro představu, proč je obtížné odhadnout vzdálenost markeru od kamery, jsem přidal obrázek 4.1, na kterém se nachází marker ve dvou vzdálenostech. Vidíme marker o rozměrech 10 × 10 cm ve vzdálenosti 5 a 6 metrů od kamery se zorným úhlem 48,5°. Můžeme si všimnout velmi malé změny velikosti markeru.



Obrázek 4.1: Porovnání markerů ve vzdálenost 5 a 6 m od kamery

Druhým problémem je malý prostor, kde lze detekovat marker. Zorné pole jedné kamery nemusí pokrýt dostatečný prostor. Může se také stát, že robot bude orientován takovým směrem, že kamerou nebude možné vidět marker.

Oba tyto problémy mohou být vyřešeny přidáním jedné nebo více kamer do systému.

Kapitola 5

Odhad polohy a rotace pomocí soustavy kamer

Problémy s přesností ve směru pohledu kamery a s pokrytím můžeme řešit přidáním jedné nebo více kamer. Tato kapitola se zabývá způsobem, jak využít data z více kamer k odhadu pózy markeru.

5.1 Definice úlohy

Úlohu definujeme jako neomezený minimalizační problém (5.1, 5.2), kde se snažíme minimalizovat chybu, kterou definujeme jako součet čtverců vzdálenosti projektovaných bodů a detekovaných bodů ve snímku z každé použité kamery. Jinými slovy hledáme posunutí a rotaci markeru, tak aby se jeho rohy po projekci do obrázků co nejvíce podobaly detekovaným rohům.

$$\min_{R,t} E(R,t) \tag{5.1}$$

$$E(R,t) \quad = \quad \sum_{j=1}^{k} \sum_{i=1}^{4} \left\| \quad proj(R_{cj}, t_{cj}, \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} P_{0i}) - Y_{ij} \right\|^2 \tag{5.2}$$

kde:
- E = chyba celkového systému
- k = počet kamer
- R_{cj} = rotační matice j-té kamery
- t_{cj} = translační vektor j-té kamery
- P_{0i} = i-tý sloupec matice P_0
- Y_{ij} = souřadnice i-tého rohu markeru na obrázku z j-té kamery
- R = hledaná rotační matice markeru
- t = hledaný vektor posunutí markeru

5.2 Popis algoritmu

Algoritmus vychází z této práce [29], je však modifikován a upraven pro více kamer. Algoritmus rozděluje problém s 6 stupni volnosti na dva problémy se 3 stupni volnosti. Oba tyto problémy jsou řešeny numericky pomocí Lavenberg-Marquardtovy iterační metody. Algoritmus využívá toho, že řešení nalezené jednou kamerou leží

blízko hledaného řešení, proto iterační metody použité pro hledaní finálního řešení dokonvergují velmi rychle.

Systém předpokládá, že všechny kamery jsou dokonale synchronizované, a také že jsou známy přesné vnitřní i vnější parametry všech využitých kamer. Dále se předpokládá, že se ve scéně nenachází žádné dva markery se stejným identifikátorem.

Marker je detekován několika kamerami (alespoň 2). Nejdříve se použije jedna kamera k odhadu polohy a rotace markeru, tento postup je popsán v kapitole 4. Tyto hodnoty slouží jako výchozí hodnoty pro iterační metody na určení polohy a rotace.

V dalším kroku je hledána poloha markeru, tedy jeho střed. Tento střed je hledán tak, aby se minimalizovala odchylka čtverců vzdáleností projekce středu a středů markeru detekovaným v obrázcích. K řešení tohoto problému je využita iterační metoda, konkrétně Levenberg-Marquardtova metoda. Jako inicializační hodnota se použije poloha odhadnutá jednou kamerou.

Potom co je nalezen střed markeru, je marker umístěn na tuto pozici a je hledána požadovaná rotace. Tentokrát se snažíme minimalizovat odchylku čtverců vzdálenosti projekce všech rohů markeru a detekovaných rohů v obrázcích. K nalezení takové rotace je také využívána stejná iterační metoda jako u odhadu polohy, inicializační rotace je převzata také z odhadu pomocí jedné kamery. Po nalezení rotace je odhad polohy a rotace markeru hotov.

5.2.1 Přehled

1. Inicializace - první kamera určí polohu a rotace markeru bez ohledu na ostatní (viz kapitola 4)

2. Nalézt střed markeru, tak aby odpovídal detekovaným středům na snímcích

3. Umístit marker do nalezeného středu

4. Nalézt odpovídající rotaci markeru tak, aby rohy markeru odpovídaly detekovaným rohům na snímcích

5. Poloha a rotace markeru je nalezena

5.2.2 Levenberg–Marquardtova metoda

Nalézt střed a rotaci markeru jsem se rozhodl numericky, pro tyto účely jsem si vybral Levenberg-Marquardtovu metodu, jejichž popis je popsán v této práci [30].

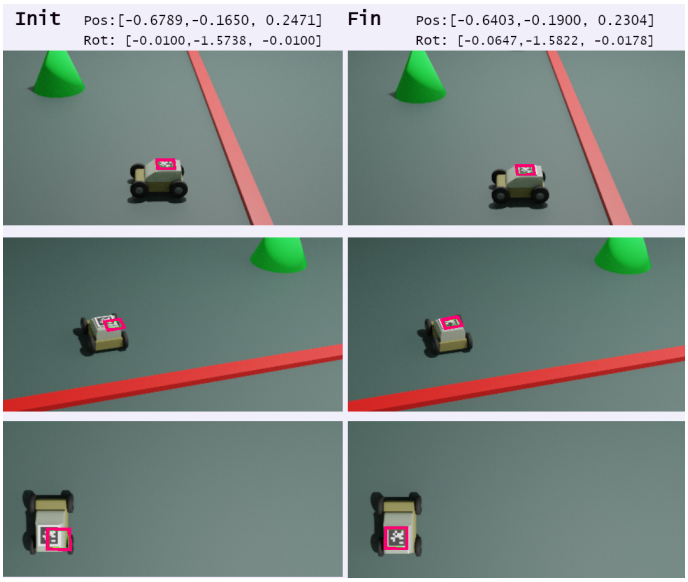
Levenberg–Marquardtova metoda je iterační metoda vhodná k řešení nelineárních nejmenších čtverců. Jedná se o kombinaci Gradientní metody a Gauss-Newtonovy metody, kombinací těchto metod se eliminují problémy Gauss-Newtonovy metody, které vznikají, když se startovací vektor nachází ve velké vzdáleností od hledaného extrému. Tato metoda je používána k řešení řady optimalizačních úloh. Je využita například pro podobnou úlohu v knihovně OpenCV a to k řešení PnP problému [28].

V OpenCV je také implementován řešič (LMSolver), který jsem využil při implementaci algoritmu. K výpočtu je potřeba k danému vstupnímu vektoru spočítat chybu a Jacobiho matici parciálních derivací.

Jacobiho matice lze získat spočtením přibližné derivace prostým odečtením a vydělením chyby ve dvou blízkých bodech.

Výpočet přibližné derivace (5.3) v bodě x, h je nějaké malé číslo a E je funkce pro výpočet chyby.

$$E'(x) = \frac{E(x-h) - E(x+h)}{2h} \tag{5.3}$$



Obrázek 5.1: Inicializovaná a vypočítaná poloha markeru zobrazená kamerami

5.3 Kalibrace multikamerového systému

Při odhadu polohy markeru s multikamerovým systémem je důležité znát přesné parametry použitých kamer, protože i malá nepřesnost v poloze nebo rotaci jedné z kamer může vést k velké chybě odhadu. Tyto parametry kamer není snadné získat, protože změřit polohy a rotace kamer je ve fyzickém světě obtížné. Proto jsem vytvořil nástroj na automatickou kalibraci polohy a rotace kamer. V této sekci popisuji použitý algoritmus.

Poloha a rotace všech kamer se automaticky spočítají z detekovaných markerů ve scéně. Všechny kamery, jejichž parametry chceme určovat, musí detekovat alespoň 3 stejné markery. Další podmínkou je, že musíme znát polohu a rotaci alespoň jedné kamery. Bez toho bychom nebyli schopni určit absolutní polohu kamer.

Hledané parametry kamer nalezeneme vyřešením minimalizační úlohy, ve které jsou hledány nejen polohy a rotace kamer, ale také polohy a rotace markerů. Základní úloha je definována takto (5.4, 5.5), kde I = počet markerů + 1.

$$\min_{R_c,T_c,T,R} \quad \sum_{m=1}^{M}\sum_{j=1}^{K} \left\| X_m \right\|^2 \tag{5.4}$$

$$X_m = \begin{cases} \sum_{i=1}^{4}(proj(R_{cj}, T_{cj}, \begin{bmatrix} R_m & T_m \\ 0 & 1 \end{bmatrix} P_{0i}) - Y_{mji}), & \text{pokud } m < I \\ proj(R_{cj}, T_{cj}, T_m) - C_{mj}, & \text{jinak} \end{cases} \tag{5.5}$$

kde:
K = počet kamer
M = počet markerů
R_{cj} = rotační matice j-té kamery
T_{cj} = translační vektor j-té kamery
T_m = střed m-tého markeru
R_m = rotace m-tého markeru
P_{0i} = i-tý sloupec matice P_0
Y_{mji} = souřadnice i-tého rohu m-tého markeru na obrázku z j-té kamery
C_{mj} = souřadnice m-středu markeru na obrázku z j-té kamery
T_c = matice se sloupci posunutí kamer
R_c = matice se sloupci rotačních vektorů kamer
T = matice se středy markerů ve sloupcích
R = matice s rotačními vektory markerů ve sloupcích
I = pořadí řešené úlohy

Tato úloha obsahuje mnoho hledaných proměnných, konkrétně $6 \times počet\_markerů \times (počet\_kamer - 1)$. To je příliš mnoho, takto by použitá LM metoda velmi obtížně konvergovala, proto je tato úloha rozdělena na dílčí úlohy. Začíná se s úlohou s méně proměnnými a postupně se pak přidávají další proměnné, dokud není vyřešena původní úloha. Definice minimalizačních úloh se navzájem liší pouze parametrem I.

V první úloze ($I = 1$) se hledají pouze středy markerů a pózy kamer. Po vyřešení této úlohy se ve druhé úloze ($I = 2$) přidá k hledaným proměnným rotace prvního markeru. Ve třetí ($I = 3$) se pak přidá k předchozím proměnným také rotace druhého markeru. Takto se pokračuje dokud nejsou nalezeny polohy a rotace všech markerů a kamer.

Jako startovací hodnoty pro minimalizační úlohy jsou použity odhadnuté polohy a rotace markerů pomocí kamery, jejichž parametry známe. Způsob určení polohy markerů je popsán v kapitole 4. Startovací hodnoty pro vnější parametry kamer jsou odhadnuty také pomocí této metody, s tím rozdílem, že se počítá naopak poloha kamery vůči markeru, jejichž hodnoty jsou odhadnuty pomocí známé kamery.

Takto je řešená kalibrace vnějších parametrů kamer v tomto systému.

Kapitola 6

Implementace lokalizačního systému

V této kapitole popisuji implementaci samotného systému, návod na použití a formát vstupních dat.

6.1 Volba použitých nástrojů

V této sekci popisuji programovací jazyk a knihovny, které jsem vybral pro tento projekt.

6.1.1 Programovací jazyk

Rozhodl jsem se naimplementovat tento lokalizační systém v jazyce C++ kvůli několika jeho výhodám. První z nich je dostupnost potřebných knihoven, konkrétně se jedná o knihovnu OpenCV obsahující mnoho užitečných funkcí pro zpracování obrazu a rekonstrukci 3D scény. Další z knihoven, které jsem potřeboval, byla knihovna pro detekci AprilTagů, která je také dostupná pro jazyk C++.

Další výhodou jazyka C++ je jeho vysoká rychlost, která je potřebná pro real-time aplikaci, jako je lokalizační systém. Knihovna OpenCV a knihovna pro detekci AprilTagů jsou také dostupné pro programovací jazyk Python, jehož rychlost běhu je nižší [31] a nemusela by být dostačující pro real-time použití.

Programovací jazyk Python jsem používal pro testování a prototypování.

6.1.2 AprilTag detektor

Pro detekci markeru typu April Tag jsem využil detektor s názvem Apriltag cpp(Windows)[32] od uživatele P-Chao. Tato knihovna vychází z jiné knihovny, a to z knihovny s názvem swatbotics/apriltags-cpp [33], ale je upravená tak, aby fungovala i na platformě Windows. Vybral jsem si tuto knihovnu, jelikož jsem chtěl, aby výsledný program fungoval jak na systémech Linux, tak na platformě Windows, na které jsem tento systém vyvíjel.

Zdrojové soubory této knihovny jsou připojeny ke zdrojovému kódu lokalizačního systému, nachází se v samostatné složce *src/april*. Knihovna obsahuje základní třídy pro detekci markeru. V obrázku nalezne všechny markery vybrané april tag rodiny.

Objekt *TagDetection* obsahuje informace o každém detekovaném markeru. Pro tento účel postačí následující informace, souřadnice všech 4 rohů markeru, jeho ID a orientace markeru. Orientace markeru značí, který z bodů je levý horní roh. Dále tento objekt nese informaci o hammingově vzdálenosti nalezeného markeru.

6.1.3 OpenCV

V projektu je použita knihovna OpenCV. Tato knihovna obsahuje řadu funkcí, např. načítání obrázku, získaní dat z webkamer a projekce 3D bodů. Také obsahuje definici matic a veškeré potřebné maticové operace (sčítání, násobení, inverze, apod.), které jsou potřeba k odhadu polohy. Vybral jsem nejnovější verzi OpenCV 4.5.0, která obsahuje také LM solver, který je využíván při iteračním výpočtu polohy markeru.

6.2 Funkce programu

V této sekci jsou popsány implementované funkce lokalizačního systému a jejich použití. Mimo odhad polohy markerů pomocí jedné nebo více kamer byl také implementován nástroj na kalibraci vnějších parametrů kamer. Dále je také možné sdílet odhadnuté pózy markerů pomocí TCP serveru dalším aplikacím.

6.2.1 Odhad polohy markerů

Program je schopný v reálném čase detekovat pomocí webkamer připojených k počítači markery ve scéně a určit jejich polohu i rotaci. Je možné pracovat s více markery ve scéně avšak za následujících podmínek. Všechny markery jsou ze stejné rodiny, mají stejné fyzické rozměry a jejich identifikátory jsou unikátní.

Ve vstupním souboru je možné nakonfigorovat konkrétní identifikátory, které systém bude detekovat, markery s ostatními identifikátory budou ignorovány. Minimální velikost detekovatelného markeru na snímku je 15 pixelů. Také lze nastavit minimální počet kamer potřebných pro odhad polohy (vychozí 1) a maximální počet iterací pro numerické výpočty (výchozí hodnota 500).

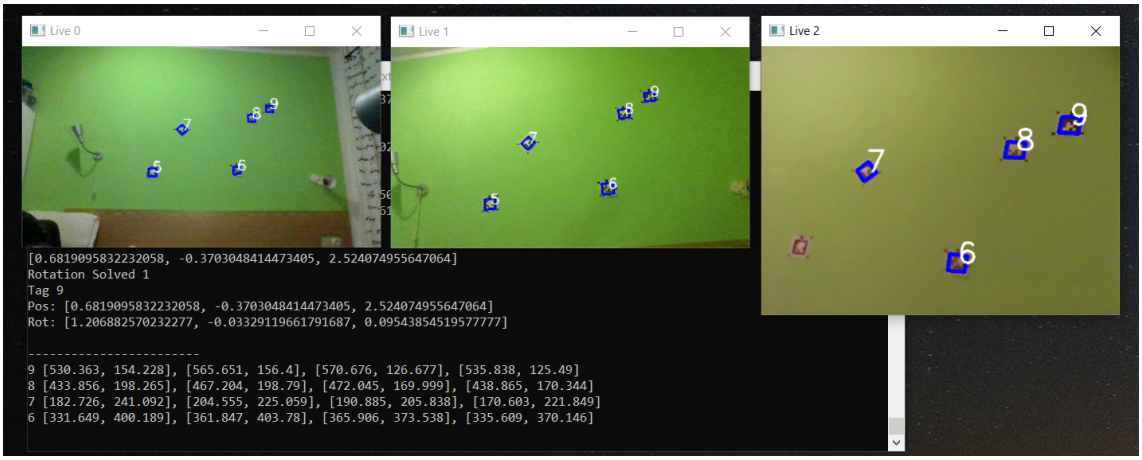
Úspěšně vypočítáné polohy a rotace markerů se vypíší na standardní výstup. Poloha středu je udávána v metrech a rotace je udávána pomocí rotačního vektoru v radiánech. Přepínačem *-euler* (viz tabulka 6.1) lze nastavit, aby se výstupní rotace převedla do eulerových úhlů (XYZ, v radiánech). Systém také zobrazuje snímky z kamer spolu se zvýrazněnými detekovanými markery, jak je vidět na obrázku 6.1.

Pro přesný odhad polohy stačí detekce markeru dvěma kamerami (viz sekce 7.1).

Velikost prostoru, ve kterém je schopen systém pracovat je popsán v sekci 7.4.

6.2.2 Kalibrace kamer

Součástí programu je nástroj na odhad polohy a rotace kamer použitých v systému. K výpočtu je potřeba detekovat alespoň 3 stejné markery všemi kamerami. Je tedy potřeba nastavit kamery tak, aby byly markery v jejich zorných úhlech. Kalibrační nástroj považuje první kameru systému jako již zkalibrovanou, její poloha a rotace se nebude měnit. Algoritmus popsaný v sekci 5.3 se automaticky spustí po



Obrázek 6.1: Snímek obrazovky se spuštěným lokalizačním systémem detekující 5 markerů pomocí 3 kamer

splnění výše uvedených podmínek. Po úspěšném skončení kalibrace jsou soubory se zjištěnými parametry uloženy do pracovní složky. Tyto soubory je možné přímo použít v lokalizačním systému.

6.2.3 Sdílení dat s jinými aplikacemi

Aby bylo možné vytvořit aplikace pracující s daty získané tímto lokalizačním systémem, program je schopen vytvořit na lokálním stroji TCP server a posílat pakety s polohami markerů všem připojeným klientům.

Odesílají se tyto hodnoty: ID markeru, poloha markeru a jeho rotace. Data se posílají v textové formě a jednotlivé hodnoty jsou odděleny středníkem. Každý paket obsahuje informaci o jednom detekovaném markeru.

6.3 Struktura programu

V této sekci jsou popsány hlavní třídy a důležité hlavičkové soubory v samotné implementaci lokalizačního systému.

6.3.1 Systemové parametetry

Třída *SystemParameters* obsahuje parametry systému převzaté ze vstupních souborů a argumentů. Mezi tyto parametry patří: velikost markerů a jejich tag rodina, vnitřní a vnější parametry použitých kamer (třída *CameraParameters*), port na kterém běží server, přepínač zda je spustěn kalibrační nástroj a další přepínače jako například maximální počet iterací a minimální počet kamer.

Tyto třídy a funkce jsou naprogramované v souborech *system_parameters.h* a *system_parameters.cpp*.

6.3.2 Detektor

Třída *Detector* obsahuje funkci *detect* pro detekci AprilTagů v obrázcích, která detekované markery uloží do výstupní proměnné. Tato výstupní proměnná je typu

mapa, kde klíč je ID markeru a hodnota je pole detekovaných markerů, kde index markeru se rovná indexu kamery, která marker detekovala.

Detekce markerů je realizována voláním funkcí knihovny pro detekci AprilTagů. Výsledky jsou uloženy v struktuře *DetectedTag*.

Implementace třídy *Detector* je implementována v souborech *detector.h* a *detector.cpp*.

6.3.3 Odhad polohy a rotace

V souborech *loc_finder.h* a *loc_finder.cpp* jsou implementovány veškeré funkce pro transformace bodů mezi souřadnicemi a odhad polohy markerů nebo kamery. Je zde funkce pro odhad jednou kamerou pomocí funkce *solvePnP*. Dále zde existuje také funkce pro odhad s více kamerami, která pak využívá numerické metody.

Je zde také definována a implementována třída *CamData*, která reprezentuje jednu kameru systému. Součástí této třídy jsou také funkce pro výpočet chyby na této kameře mezi promítnutými body a detekovanými body, které se pak využívají v numerických metodách.

6.3.4 Numerické výpočty

Numerické výpočty pro odhad polohy markerů, odhad rotace markerů a pro kalibrace vnějších parametrů kamer jsou realizovány pomocí LM iterační metody. K řešení je využit *LMSolver* z knihovny OpenCV. K použití tohoto solveru je potřeba předat implementaci *LMSolver::Callback*, která obsahuje funkci *compute*.

Pro každý výpočet je implementován callback v samostatném hlavičkovém souboru. Soubor *position_lm_callback.h* implementuje callback pro výpočet středu markeru (viz krok 2 v 5.2.1). Soubor *rotation_lm_callback.h* obsahuje callback pro výpočet rotace markeru po tom, co známe jeho střed (viz krok 4 v 5.2.1). Implementace LM metody pro kalibraci vnějších parametrů kamer, která je popsaná v kapitole 5.3, se nachází v souborech *calib_lm_callback.h* a *multi_tag_calib_lm_callback.h*

Počet maximálních iterací pro odhad polohy a rotace je možné nastavit pomocí přepínače *-max_iter* (v tabulka 6.1).

6.3.5 TCP Server

Jednoduchý TCP server je implementován v souborech *ls_server.h* a *ls_server.cpp*. Je zde napsán kód pro systémy Windows a Linux. Pomocí ifdef je realizováno přepínání mezi těmito implementacemi.

Server každému připojenému klientovi nejdříve napíše uvítací zprávu. Při každé iteraci systému jsou odeslány pakety s úspěšně vypočítanými polohami a rotacemi markerů. Jeden paket obsahuje data popisující jeden marker, pošle se tedy vždy počet paketů rovnající se počtu úspěšně odhadnutých markerů.

6.3.6 Main

V souboru *main.cpp* lze nalézt funkci main a funkce pro spuštění webkamer a volání metod pro určení polohy markerů v závislosti na počtu kamer, které ho detekují. Spuštění kamer je realizováno pomocí funkcí z knihovny OpenCV.

6.4 Vstupní argumenty a vstupní soubory

Lokalizační systém přebírá vstupní parametry ze vstupních souborů a z přepínačů. Seznam přepínačů a jejich funkcí je vypsán v tabulce 6.1.

Tabulka 6.1: Seznam přepínačů s jejich funkcemi

Přepínač	Parametr	Funkce
-dc	cesta k souboru	připojení seznamu webkamer
-d	cesta k souboru	připojení seznamu kamer a vstupních obrázků
-ts	velikost markeru [m]	zadání velikosti markeru
-tf	tag rodina	výběr použité tag rodiny
-m	cesta k souboru	připojení souboru s parametry markerů
-server	číslo portu	spustí TCP server na vybraném portu
-e	-	spustí kalibrační nástroj
-euler	-	výstupní rotace v eulerových úhlech (XYZ)
-help	-	vypíše použití programu
-max_iter	počet iterací	nastavení maximální počet iterací
-min_cam	počet kamer	minimální počet kamer pro odhad
-max_cam	počet kamer	maximální počet kamer pro odhad
-print_times	-	vypíše časy trvání jednotlivých procedur

Parametry kamer a markerů jsou programu předávány pomocí textových souborů s jednoduchou syntaxí. Na každém řádku se nachází jeden parametr.

6.4.1 Soubor s parametry kamery

Tento soubor obsahuje vnitřní a vnější parametry jednotlivých kamer. Parametry jsou zapsány v souboru následovně: ohniskové vzdálenosti (v pixelech), optický střed (v pixelech), parametry zkreslení, translační vektor (světové souřadnice), rotační vektor (světové souřadnice) a požadované rozlišení kamery.

6.4.2 Soubor se seznamem kamer

V tomto souboru jsou uložena identifikační čísla kamer, které uživatel hodlá využít. K jednotlivým kamerám je přiřazena adresa k souboru s parametry kamery (6.4.1).

6.4.3 Soubor se seznamem markerů

Systému lze předložit seznam markerů, které uživatel hodlá detekovat. Pokud se detekují markery s ID, které se nenachází v seznamu, marker bude ignorován a nespustí se odhad polohy. Pokud uživatel nezadá žádné ID, jsou detekovány všechny markery z vybrané rodiny.

Tento seznam je uložen v tomto souboru spolu s definicí tag rodiny a fyzickou velikostí markeru. Tudíž může nahradit přepínače *-tf* a *-ts*. Soubor se připojí k volání programu pomocí přepínače *-m*.

Podrobná syntaxe souboru je popsána v soubrou *README.md* v příloze.

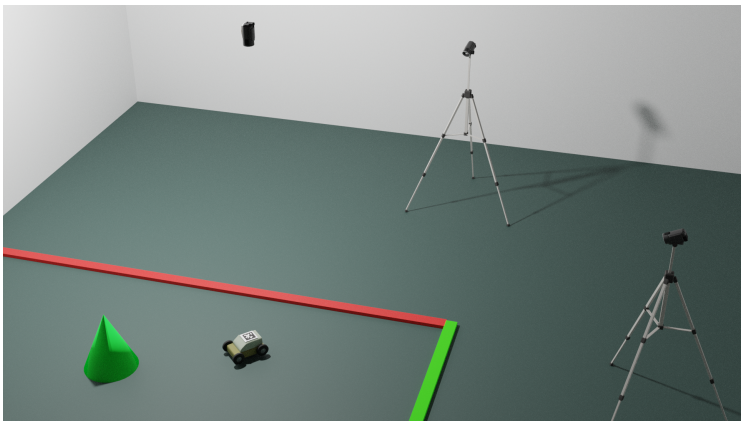
Kapitola 7

Experimentální ověření vlastností systému

V této kapitole popíši 4 experimenty, které jsem provedl za účelem zjistit vlastnosti implementace lokalizačního systému. Zaměřil jsem se na určení přesnosti systému,jeho časové náročnosti a velikost pokrytého prostoru.

7.1 Přesnost za použití 1 vs 2 vs 3 kamer

V tomto experimentu jsem chtěl otestovat základní myšlenku tohoto projektu, že použitím více kamer, které snímají scénu z různých úhlů, se zvýší přesnost systému. Otestoval jsem tedy systém s jednou kamerou, pak se dvěma a nakonec se třemi kamerami. Následně jsem porovnal jejich přesnost.



Obrázek 7.1: Scéna virtuální laboratoře se 3 kamerami

7.1.1 Popis experimentu

Experiment jsem provedl ve virtuálním prostředí vymodelovaném v programu Blender [34]. Provádět experiment ve virtuálním prostředí přináší řadu výhod, největší z nich je to, že všechna vstupní a výstupní data lze zjistit naprosto přesně, tudíž měření nebude zkresleno chybou ze vstupních dat. Další výhodou je to, že lze použít mnoho kamer, které lze umístit kamkoliv do prostoru.

Virtuální laboratoř obsahovala 3 kamery (obr. 7.1), první kamera snímala prostor zepředu, druhá z boku a třetí ze shora. Všechny kamery renderovaly snímky v rozlišení 1920×1080, FoV kamer bylo $48,5°$ a zkreslení kamer bylo nulové. Maximální počet iterací systému pro odhad polohy a rotace markerů byl nastaven 80000 tak, aby bylo jisté, že se všechny polohy a rotace vypočítají. Marker o velikosti 10 cm byl umístěn na vrchní části robota.

Postupně jsem umístil robota do 10 poloh, vyrenderoval jsem snímky, které jsem vložil jako vstup do navrhovaného lokalizačního systému. Poté jsem porovnal spočítané hodnoty se skutečnými. Testovací polohy markeru se nacházely v rozmezí 1,5 m - 3,5 m od nejbližší kamery.

Měřil jsem také chybu vypočítané rotace od skutečné. Chybu jsem měřil v každé složce eulerových úhlů jako rozdíl naměřené hodnoty od skutečné.

Tabulka 7.1: Výsledky měření. Byla měřena euklidovská vzdálenost vypočítané polohy od skutečné při použití různého počtu kamer. Hodnoty jsou uvedeny centimetrech.

Test	1 Kamera	2 Kamery	3 Kamery
1	8,43	0,59	0,57
2	11,90	0,41	0,53
3	8,58	0,44	0,53
4	6,73	0,63	0,60
5	11,45	0,36	0,52
6	15,52	0,42	0,50
7	11,49	0,64	0,64
8	7,94	0,58	0,59
9	12,12	0,39	0,51
10	9,84	0,45	0,55
Průměr	10,401	0,491	0,553
Sm. odchylka	2,465	0,102	0,043

Tabulka 7.2: Výsledky měření. Průměrná chyba a směrodatná odchylka jedné vypočítané složky eulerových úhlů od skutečné hodnoty pro různý počet kamer.

	1 Kamera	2 Kamery	3 Kamery
Průměr	0,116°	0,116°	0,116°
Sm. odchylka	0,099°	0,099°	0,099°

7.1.2 Výsledky

V tabulce 7.1 najdeme výsledné hodnoty v testu přesnosti určení polohy. Hodnoty udávají absolutní odchylku vypočítané polohy markeru od jeho skutečné polohy.

Z naměřených hodnot vidíme, že se použitím 2 kamer výrazně zvýšila přesnost systému. Průměrná odchylka dosahovala v tomto případě hodnoty 0,491 cm, což je výrazně méně než za použití jedné kamery (necelých 10 cm).

Přidáním třetí kamery se průměrná chyba prakticky nezměnila, snížila se pouze směrodatná odchylka. To lze vysvětlit skutečností, že iterační algoritmus skončí, pokud součet chyb projekcí oproti detekovaným bodům klesne pod určitou mez.

Je tedy pravděpodobné, že této meze bylo dosaženo už za použití 2 kamer. Toto podporuje i to, že směrodatná odchylka je velmi nízká a to 1 mm.

Lze tedy usuzovat, že přidáním čtvrté či dalších kamer se přesnost systému už nezvýší. Pro praktické použití tedy stačí, když bude marker zachycen 2 kamerami, jelikož algoritmus dokonverguje s větším počtem kamer výrazně pomaleji (7.3) a zvýšení přesnosti není prakticky žádné. Hlavní výhoda více kamer než 2 spočívá v pokrytí většího prostoru a zachycení markeru z jiných úhlů pro případ, že ho jiné kamery nezachytí.

Naměřené chyby v odhadech rotace nalezneme v tabulce 7.2. V tabulce se nachází průměry chyb ve všech složkách rotace. Chyba byla velmi nízká a stejná pro všechny počty kamer. Stejné výsledky lze vysvětlit tím, že jako startovací odhad rotace se používá odhad podle první kamery. Odhad byl ve všech případech dostatečně přesný, proto ho nebylo třeba korektovat pomocí iteračních metod.

7.2 Přesnost systému s fyzickými kamerami

Cílem toho experimentu bylo ověřit přesnost systému při reálném použití s fyzickými kamerami. Umístil jsem markery do scény a pomocí implementovaného systému jsem spočítal vzdálenost mezi markery. Poté jsem spočítáná data porovnal s naměřenými vzdálenosti. Zvolil jsem tento způsob, jelikož ho bylo možné snadněji realizovat narozdíl od měření odchylky v každé ose. Vypovídací hodnota experimentu by měla být stále dostačující.

Experiment měl 2 části, v první se markery nacházely v 3D prostoru, zatím co v druhé byly markery umístěny do jedné roviny. Na obrázku 7.2 lze vidět průběh měření obou částí.

Obrázek 7.2: Vlevo foto z první části. Vpravo foto z druhé části s číslama markerů a s naznačenými měřenými vzdálenostmi

7.2.1 Popis experimentu

V první části se markery umisťovaly volně do 3D prostoru o velikosti $2 \times 1,5 \times 1$ m^3. Byly využity 2 markery, první zůstal pevně na jednom místě, zatím co druhý marker byk postupně umisťován do 20 poloh v testovacím prostoru. Vzdálenost markerů od nejbližší kamery se pohybovala v rozmezí $0,95 - 2$ m.

Ve druhém experimentu bylo 5 markerů umístěno do jedné roviny (obr. 7.2) a měřily se vzájemné vzdálenosti mezi všemi markerami.

Při obou experimentech jsem použil 2 kamery s rozlišením 1280×720. Použít pouze 2 kamery jsem se rozhodl na základě výsledků experimentu popsaném v sekci

7.1. Detekoval jsem markery o velikosti 71 mm z rodiny *Tag16h5*. Ke kalibraci kamer jsem využil implementovaný kalibrační nástroj, který polohy kamer vypočítal ze 4 markerů umístěných do 3D prostoru pro první experiment a pro druhý do stejné roviny jako se nacházely markery při druhém experimentu. Systém byl nastaven přepínačem *min-cam*, tak aby se pro odhad vždy použily 2 kamery. Pro polohu markerů byl použit průměr z 20 odhadů. Poté se spočítala vzdálenost mezi oběma průměrnými polohami. Vzdálenosti se počítaly s přesností na milimetry.

Tabulka 7.3: Výsledky první části. Porovnávala se spočítaná (odhadnutá programem) a naměřená vzdálenost mezi dvěma markery. Ve druhém a třetím sloupečku je vzdálenost od první a druhé kamery. Všechny hodnoty jsou uvedeny v centimetrech.

Test	Vzd. kam. 1	Vzd. kam. 2	Naměřená	Odhadnutá	Chyba
1	135,8	161,2	90,6	90,9	0,3
2	143,8	142,2	102,1	102,6	0,5
3	192,8	183,3	68,3	68,0	0,3
4	100,1	95,7	150,7	149,3	1,4
5	193,4	227,0	38,2	38,8	0,6
6	174,7	145,2	111,5	112,6	1,1
7	111,4	98,2	144,1	145,1	1,0
8	169,0	203,5	55,3	56,2	0,9
9	118,5	135,7	119,0	119,2	0,2
10	212,1	168,1	132,7	131,6	1,1
11	209,9	165,7	149,5	149,1	0,4
12	154,2	148,3	103,1	104,5	1,4
13	103,2	108,7	135,8	136,2	0,4
14	190,6	194,5	65,8	66,4	0,6
15	180,2	166,3	109,9	111,4	1,5
16	100,3	95,2	151,0	151,9	0,9
17	147,8	175,5	71,8	72,8	1,0
18	149,8	132,8	123,4	124,1	0,7
19	141,7	135,4	111,8	112,6	0,8
20	228,4	186,7	148,7	148,8	0,1
Průměr					0,76
Sm. odchylka					0,41

7.2.2 Výsledky

Výsledky první části experimentu se nachází v tabulce 7.3. Průměrná chyba dosahovala velikosti $0,76$ cm pro vzdálenosti v rozmezí $55 - 151$ cm. Velikost průměrné chyby lze vysvětlit menším rozlišením použitých kamer a možnou nepřesností ve vnitřních a vnějších parametrech kamer, které byly odhadnuty pomocí kalibrační šachovnice a kalibračního nástroje. Nicméně tuto přesnost považuji za dostatečnou pro účely tohoto lokalizačního systému.

Velikost chyby v tomto experimentu byla v prostoru různě velká. Avšak chyba nezávisela na vzdálenosti od kamery, protože markery byly umisťovány v poměrně malém rozsahu vzdáleností od kamer (od 1 m do 2 m). Na takovou vzdálenost se nižší přesnost vzhledem k větší vzdálenosti od kamery příliš neprojevila. Největší

vliv na velikost chyby však mělo pravděpodobě umístění vzhledem k místům, kde se nacházely kalibrační markery. To například dokazuje test číslo 4, kde se marker nácházel velmi blizko kamer, ale i přesto byla chyba velká (1,4 cm). Žádný z kalibračních markerů se totiž nenacházel poblíž. Opačný případ je test číslo 20, kde se marker nacházel nejdále od kamer, ale oba markery ležely ve stejné rovině jako dva z kalibračních markerů. Díky tomu byla chyba pouze 0,1 cm.

Toto je důsledek nedokonalé kalibrace kamer. v ideálním případě by největší vliv na velikost chyby měla vzdálenost od kamery. I toto dokazuje, jak velký vliv na výsledky má správná kalibrace kamer.

Naměřené hodnoty z druhého experimentu lze nalézt v tabulce 7.4. Průměrná chyba dosahovala $0,49$ cm, což je srovnatelné s první částí, jelikož v druhé části se vzdálenosti pohybovaly pouze od 20 cm do 124 cm.

Pokud srovnáme výsledky virtuálního experimentu 7.1 a experimentu s fyzickými kamerami, výsledky jsou podobné. Experiment s fyzickými kamerami byl měřen na menším prostoru, což by mělo snižovat průměrnou chybu. Ale na druhou stranu se použily kamery s menším rozlišením (FullHD v prvním experimentu a HD v druhém) a menší markery (100 mm v prvním a ve druhém 71 mm). Dále poloha a rotace kamer byla zatížena chybou, jelikož tyto hodnoty byly pouze odhadnuty pomocí kalibračního nástroje. Průměrná chyba odhadu v prvním experimentu byla 0,491 cm a ve druhém 0,76 cm.

Systém se navíc během experimentu ukázal jako prakticky použitelný. S využitím dvou levných webkamer dosahoval dostatečné přesnosti. Pomocí implementovaného kalibračního nástroje se snadno spočítaly vnější parametry kamer. Systém odhadnul 20 poloh markerů (ze kterých se počítal průměr) velmi rychle a to během $2 - 3$ sekund pro každé umístění.

Tabulka 7.4: Výsledky druhé části. Porovnávala se spočítaná (odhadnutá programem) a naměřené vzdálenost mezi dvěma markerami. Hodnoty jsou uvedeny centimetrech.

Markery	Naměřená	Odhadnutá	Chyba
(1, 5)	46,0	45,1	0,9
(1, 6)	59,0	59,0	0,0
(1, 8)	64,9	64,6	0,3
(1, 9)	84,9	85,2	0,3
(5, 6)	73,7	72,9	0,8
(5, 8)	103,3	102,6	0,7
(5, 9)	124,0	123,7	0,3
(6, 8)	51,2	51,9	0,7
(6, 9)	67,9	68,6	0,7
(8, 9)	20,6	20,8	0,2
Průměr			0,49
Sm. odchylka			0,28

7.3 Časová náročnost implementace

Cílem tohoto experimentu je změřit časovou náročnost běhu implementovaného lokalizačního systému v závislosti na počtu kamer a počtu markerů.

7.3.1 Popis experimentu

Experiment jsem prováděl na svém počítači se 3 fyzickými webkamerami s rozlišením 1280×720, 1280×720 a 640×480. Počítač, na kterém byl spuštěn program měl následující parametry: Intel Core i7 1065G7 Ice Lake 1,3-3,90 GHz 4 jádra HT (8 vláken) a 16 GB RAM. Kamery byly zkalibrovány lokalizačním systémem. Měřil jsem průměrné časy běhu systému s 1, 2 a nakonec se 3 kamerami. Nejdříve scéna neobsahovala žádné markery. Poté byly po jednom přidávány markery až do počtu 5. Jednalo se o markery velikosti 72 mm z rodiny Tag16h5. Systém vždy pracovat přibližně 30 sekund, během kterých byly markery umisťovány do různých pozic s různými orientacemi. Z časů byly posléze vypočítány průměrné časy pro jednu iteraci běhu systému.

Během času, kdy lokalizační systém pracoval, byly měřeny časy dvou činností, které konzumují nejvíce výpočetního výkonu, detekce markerů a výpočet polohy a rotace markerů. Dále byl měřen průměrný počet snímků za sekundu.

Tabulka 7.5: Průměrné časy a FPS při použití 1 kamery 1280×720

Počet markerů	0	1	2	4	3	5
Detekce [ms]	55,3	53,6	55,2	56,7	57,5	58,2
Výpočet poloh [ms]	-	0,7	1,5	1,8	2,0	2,5
Iterace celkem [ms]	101,6	102,6	102,8	103,0	103,0	103,3
FPS	9,84	9,75	9,73	9,71	9,70	9,68

Tabulka 7.6: Průměrné časy a FPS při použití 2 kamer s rozlišením 1280×720

Počet markerů	0	1	2	4	3	5
Detekce [ms]	86,7	84,9	85,1	85,3	92,1	98,2
Výpočet poloh [ms]	-	1,7	2,4	3,3	3,6	4,5
Iterace celkem [ms]	102,0	105,7	111,7	126,3	136,4	145,7
FPS	9,80	9,45	8,95	7,91	7,33	6,86

Tabulka 7.7: Průměrné časy a FPS při použití 2 kamer s rozlišením 1280×720 a 1 s rozlišením 640×480

Počet markerů	0	1	2	4	3	5
Detekce [ms]	93,6	98,4	103,4	104,5	113,3	116,6
Výpočet poloh [ms]	-	16,9	41,4	60,7	100,9	128,6
Iterace celkem [ms]	102,4	127,5	115,9	159,8	220,5	255,5
FPS	9,76	7,84	6,58	6,25	4,53	3,91

7.3.2 Výsledky

V tabulce 7.5 se nachází výsledky měření za použití jedné kamery. V tomto případě se nevyužívá algoritmu pro odhad polohy markeru z více kamer, který využívá náročné iterační metody, proto jsou časy nízké. Zvýšení počtu markerů nehraje příliš velkou roli, počet snímků za sekundu zůstává stabilně na hodnotě 9-10.

Výsledky měření při použití 2 a 3 kamer lze nalézt v tabulce 7.6 a 7.7.

Čas potřebný pro výpočet polohy markerů se může velmi lišit, jelikož k určení polohy markeru více kamerami se využívají iterační algoritmy. V některých případech algoritmus skončí po 1 iteraci, v jiných algoritmus nedokonverguje a skončí po dosažení limitu 500 iterací.

Čas detekce markerů zůstával stabilní díky tomu, že detekce markerů zachycených jednotlivými kamerami je vyhodnocována paralelně. Paralelní odhad poloh a rotace bohužel není zatím naimplementován.

7.4 Velikost pokrytého prostoru

Velikost prostoru, který je schopen systém sledovat, záleží na počtu použitých kamer, na jejich rozmístění, rozlišení a FoV. Velikost prostoru lze odhadnout pomocí maximální vzdálenosti, ve které je marker detekovatelný.

Maximální vzdálenost, na kterou je schopná kamera detekovat markeru záleží především na rozlišení a FoV kamery a také na velikosti a rodině markeru. Minimální velikost detekovatelného markeru na snímku je 15 pixelů. Z toho lze pomocí vzorečku (7.1) odhadnout přibližnou maximální vzdálenost, ve které je marker detekovatelný. Ve vzorečku se zanedbává zkreslení a další podmínky, které detektor testuje. Dále předpokládáme, že marker je natočen čelem ke kameře. Výsledek tohoto vzorečku slouží jako horní orientační vzdálenost.

$$max_d = \frac{width \cdot t_s}{2 \cdot 15 \cdot \tan \frac{FoV}{2}} \tag{7.1}$$

kde: max_d = maximální vzdálenost pro detekci markeru v metrech
$width$ = šířka snímku v pixelech
t_s = velikost markera v metrech
FoV = zorný úhel použité kamery

Kamera s rozlišením 1280×720 pixelů s FoV 74° byla schopná při testu se skutečnou kamerou detekovat marker (Tag16h5) o velikosti 72 mm maximálně ve vzdálenosti 388 cm. Pokud dosadíme tyto hodnoty do rovnice (7.1) vyjde nám maximální vzdálenost $407,7$ cm, což je o 20 cm méně, než jsem naměřil.

7.4.1 Přesnost v závislosti na vzdálenosti

Provedl jsem malý experiment pomocí jedné kamery se stejnými parametry jako jsou uvedeny výše. Měřil jsme pevně danou vzdálenost mezi dvěma markery umístěnými na tyči ve vzdálenostech 60, 150, 250 a 370 cm od kamery. Poslední vzdálenost byla vybrána jako největší, kde byly markery stabilně detekovány. Markery (Tag16h5) o velikosti 72 mm byly na tyči umístěny $51,2$ cm od sebe. Poté jsem přidal další kameru umístěnou paralelně s první kamerou a opakoval jsem měření. Tato kamera dosahovala rozlišení také 1280×720 pixelů a její FoV byl 78°.

V tabulce 7.8 nalezneme výsledky měření. Vídíme závislost velikosti chyby na vzdálenosti od kamer. Při posledním měření ve vzdálenosti 370 cm, která byla pro tyto kamery a marker na hranici detekovatelnosti, chyba dosahovala 3 cm pro jednu kameru a 2 cm pro dvě, což jsou poměrně velké hodnoty. Na druhou stranu je možné

Tabulka 7.8: Chyba v závislosti na vzdálenost při použití jedné a dvou kamer. Hodnoty jsou v centimetrech.

Vzdálenost	1 kamera		2 kamery	
	Odhadnutá	Chyba	Odhadnutá	Chyba
60	51,4	0,2	51,0	0,2
150	50,8	0,4	50,9	0,3
250	52,3	1,2	52,0	0,8
370	54,2	3,0	53,2	2,0

usuzovat, že tyto hodnoty jsou maximální velikostí chyb, jelikož vzdálenější marker už nelze detekovat.

Až na krajní případy lze tedy říci, že systém je použitelný vždy, pokud je marker detekován. Záleží však, jak velká chyba je uživatelem tolerována.

Pokud nám vyhovuje maximální velikost chyby ze zmíněného experimentu, tak lze odhadnout velikost pokryté oblasti danou kamerou s markerem přibližně na $4 \times 2,25\ m^2$.

Velikost pokrytého prostoru lze zvýšit použitím větších markerů a kamery s větším rozlišením. Například při použití kamery s rozlišením 1920×1080 pixelů, FoV 74° a markeru o velikosti 12 cm, lze odhadnout maximální detekovatelnou vzdálenost na 9,8 m. Dalo by se touto kamerou bezpečně pokrýt plocha o velikosti přibližně $14 \times 8\ m^2$.

Závěr

Cílem práce bylo navrhnout a implementovat lokalizační systém pro mobilní roboty, který by byl schopen určit polohu a rotaci robotů. Další cílem také bylo určit vlastnosti implementovaného systému. Tyto cíle byly splněny.

V prvních kapitolách jsem krátce popsal některé existující lokalizační systém a teoretický základ potřebný pro implementaci systému. V další kapitolách jsem již popisoval mnou navržený lokalizační systém, který využívá vizuální značky a jednu či více kamer k určení polohy a rotace robotů. Navrhovaný systém jsem implementoval a posléze i otestoval jeho přesnost a rychlost ve 3 experimentech.

V simulaci se chyba odhadu rotace pohybovala pod 1° ve všech testech. V experimentu se dvěma levnými webkamerami systém dosahoval při odhadu polohy průměrné chyby $0,76$ cm a maximální $1,5$ cm při měření délek od 30 cm do 2 m ve vzdálenosti od 95 cm do 2 metrů od nejbližší kamery. Hodnoty FPS během detekce jednoho markeru dvěma kamerami se pohyboval mezi 9 a 10. Přesnost a rychlost systému se v experimentech ukázaly jako plně dostačující pro účely tohoto projektu.

Implementovaný systém dokáže poskytnout dobrý odhad polohy a rotace markeru i s méně kvalitními levnými webkamerami. Pracuje v reálném čase a vypočítané hodnoty snadno sdílí dalším aplikacím skrze TCP server. Kalibrační nástroj poskytuje použitelné odhad vnější polohy použitých kamer. Systém byl navržen především pro lokalizaci mobilních robotů, ale lze ho využít kdekoliv jinde, kde je možné na sledovaný objekt umístit vizuální marker.

Prostor pro zlepšení vidím především v rozvinutí nástroje pro kalibrace vnější parametrů kamer, jelikož přesná kalibrace kamer je klíčová pro správné fungování systému. Pro větší počet kamer totiž nástroji trvá delší dobu než nalezne dostačující odhad.

Dalším nápadem na zlepšení je rozšíření serveru, tak aby byl schopen přijímat informace o detekovaných markerech z jiných zařízení. Další zařízení by pomocí svých kamer detekovaly markery a na server by odeslaly pouze informace o detekovaných markerech. Na serveru by běžel tento systém a pomocí dat ze všech zařízení by spočítal polohu a rotaci markeru. Tímto by se zlepšila rychlost a škálovatelnost systému.

Bibliografie

1. MEJJAIL; NITSCHE M.; T. KRAJNÍK; P. ČÍŽEK; M.; DUCKETT, T. WhyCon: An Efficient,Marker-Based Localization System. *IROS 2015: Proceedings IEEE/RSJ International Conference on Inteligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems, Hamburg*. 2015. Dostupné také z: http://eprints.lincoln.ac.uk/id/eprint/18877/1/2015_irososar_whycon.pdf.
2. ULRICH, Jirı; LIGHTBODY, Peter; WEINSTEIN, Aaron; MAJER, Filip; KRAJNIK, Tomáš. WhyCode: Efficient and Versatile Fiducial Localisation System. [N.d.].
3. KRAJNÍK, Tomáš; NITSCHE, Matías; FAIGL, Jan; VANĚK, Petr; SASKA, Martin; PŘEUČIL, Libor; DUCKETT, Tom; MEJAIL, Marta. A Practical Multirobot Localization System. *Journal of Intelligent & Robotic Systems*. 2014. ISSN 0921-0296. Dostupné z DOI: 10.1007/s10846-014-0041-x.
4. PIVOŇKA, Tomáš. Vizuální lokalizace pro experimentaci v mobilní robotice. *bachelor thesis, Czech Technical University in Prague*. 2016.
5. *Vicon*. Dostupné také z: https://www.vicon.com/.
6. *Vjcon Sport Performance*. Dostupné také z: https://www.vicon.com/applications/life-sciences/sports-performance/.
7. MERRIAUX, Pierre; DUPUIS, Yohan; BOUTTEAU, Rémi; VASSEUR, Pascal; SAVATIER, Xavier. A study of vicon system positioning performance. *Sensors*. 2017, roč. 17, č. 7, s. 1591.
8. VAN DER AA, Nico; LUO, X; GIEZEMAN, Geert-jan; TAN, Robby; VELTKAMP, Remco. Utrecht Multi-Person Motion (UMPM) benchmark. 2012.
9. BABINEC, Andrej; JURIŠICA, Ladislav; HUBINSKÝ, Peter; DUCHOŇ, František. Visual Localization of Mobile Robot Using Artificial Markers. *c.* 2014, roč. 96. Dostupné z DOI: 10.1016/j.proeng.2014.12.091.
10. FIALA, M. ARTag, a fiducial marker system using digital techniques. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005, sv. 2, 590–596 vol. 2. Dostupné z DOI: 10.1109/CVPR.2005.74.
11. *ARToolKit*. Dostupné také z: http://www.hitl.washington.edu/artoolkit/.
12. *Aruco*. Dostupné také z: http://www.uco.es/investiga/grupos/ava/node/26.
13. OLSON, Edwin. AprilTag: A robust and flexible visual fiducial system. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, s. 3400–3407.

14. *CMG LEE*. Dostupné také z: https://commons.wikimedia.org/wiki/File:Comparison_of_augmented_reality_fiducial_markers.svg.

15. KOEDA, Masanao; YANO, Daiki; SHINTAKU, Naoki; ONISHI, Katsuhiko; NOBORIO, Hiroshi. Development of Wireless Surgical Knife Attachment with Proximity Indicators Using ArUco Marker. In: 2018, s. 14–26. ISBN 978-3-319-91243-1. Dostupné z DOI: 10.1007/978-3-319-91244-8_2.

16. SANI, M. F.; KARIMIAN, G. Automatic navigation and landing of an indoor AR. drone quadrotor using ArUco marker and inertial sensors. In: *2017 International Conference on Computer and Drone Applications (IConDA)*. 2017, s. 102–107. Dostupné z DOI: 10.1109/ICONDA.2017.8270408.

17. WESTMAN, Eric; KAESS, Michael. Underwater AprilTag SLAM and calibration for high precision robot localization. *tech. rep.* 2018.

18. ZHENGLONG, Guo; QIANG, Fu; QUAN, Quan. Pose Estimation for Multicopters Based on Monocular Vision and AprilTag. In: *2018 37th Chinese Control Conference (CCC)*. 2018, s. 4717–4722.

19. CHEN, Jianwei; GAO, Yue; LI, Shaoyuan. Real-time Apriltag Inertial Fusion Localization for Large Indoor Navigation. In: *2020 Chinese Automation Congress (CAC)*. 2020, s. 6912–6916.

20. SANTOS CESAR, Diego Brito dos; GAUDIG, Christopher; FRITSCHE, Martin; REIS, Marco A dos; KIRCHNER, Frank. An evaluation of artificial fiducial markers in underwater environments. In: *OCEANS 2015-Genova*. 2015, s. 1–6.

21. *AprilTag GitHUb*. Dostupné také z: https://github.com/AprilRobotics/apriltag.

22. *TagDetect — rc_visard21.01.1documentation*. Dostupné také z: https://doc.rc-visard.com/latest/en/tagdetect.html.

23. *AprilTag User Guide*. Dostupné také z: https://github.com/AprilRobotics/apriltag/wiki/AprilTag-User-Guide#choosing-a-tag-family.

24. KOLEČKÁŘ, David. Localization System for a Multi-robot System. *bachelor thesis, Czech Technical University in Prague*. 2017.

25. *Camera Calibration and 3D Reconstruction*. Dostupné také z: https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#details.

26. *pcigeomatics.com*. Dostupné také z: https://www.pcigeomatics.com/geomatica-help/concepts/orthoengine_c/Chapter_45.html.

27. *Camera Calibration*. Dostupné také z: https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html.

28. *OpenCV solvepnp.cpp*. Dostupné také z: https://github.com/opencv/opencv/blob/fc1a15626226609babd128e043cf7c4e32f567ca/modules/calib3d/src/solvepnp.cpp.

29. REGAS, Antonio Badal. *3D Pose Estimation of Visual Markers* [A Degree Thesis Submitted to ETSETB (UPC), TU Wien]. 2015.

30. RANGANATHAN, Ananth. The levenberg-marquardt algorithm. *Tutoral on LM algorithm*. 2004, roč. 11, č. 1, s. 101–110.

31. PRECHELT, Lutz. An empirical comparison of c, c++, java, perl, python, rexx and tcl. *IEEE Computer*. 2000, roč. 33, č. 10, s. 23–29.

32. *P.Chao apriltags-cpp(Windows)*. Dostupné také z: https://github.com/P-Chao/apriltags-cpp-win.

33. *Swatbotics apriltags-cpp*. Dostupné také z: https://github.com/swatbotics/apriltags-cpp.

34. *Blender*. Dostupné také z: https://www.blender.org/.

Seznam obrázků

1 Roboty s vizuálními značkami 2
2 Soustava kamer snímající roboty 2

1.1 Lokalizační systém WhyCon 3
1.2 Kruhový marker WhyCon 4
1.3 Vicon kalibrační hůlka s 5 markery 5

2.1 Porovnání markerů různých typů 7
2.2 Model robota s umístěným AprilTag markerem 8
2.3 Různé Tag Family April Tagů 9

3.1 Model dírkové kamery s popisem souřadnicových systémů 12
3.2 Ukázka radiálního zkreslení 14
3.3 Kalibrační šachovnice sloužící pro zjištění parametrů kamery 16

4.1 Porovnání markerů ve vzdálenost 5 a 6 m od kamery 19

5.1 Inicializovaná a vypočítaná poloha markeru zobrazená kamerami . . . 23

6.1 Snímek obrazovky se spuštěným lokalizačním systémem detekující 5 markerů pomocí 3 kamer 27

7.1 Scéna virtuální laboratoře se 3 kamerami 31
7.2 Vlevo foto z první části. Vpravo foto z druhé části s číslama markerů a s naznačenými měřenými vzdálenostmi 33

Seznam tabulek

2.1 Počet tagů pro různé rodiny 10

6.1 Seznam přepínačů s jejich funkcemi 29

7.1 Výsledky měření. Byla měřena euklidovská vzdálenost vypočítané polohy od skutečné při použití různého počtu kamer. Hodnoty jsou uvedeny centimetrech. 32

7.2 Výsledky měření. Průměrná chyba a směrodatná odchylka jedné vypočítané složky eulerových úhlů od skutečné hodnoty pro různý počet kamer. 32

7.3 Výsledky první části. Porovnávala se spočítaná (odhadnutá programem) a naměřená vzdálenost mezi dvěma markery. Ve druhém a třetím sloupečku je vzdálenost od první a druhé kamery. Všechny hodnoty jsou uvedeny v centimetrech. 34

7.4 Výsledky druhé části. Porovnávala se spočítaná (odhadnutá programem) a naměřené vzdálenost mezi dvěma markerami. Hodnoty jsou uvedeny centimetrech. 35

7.5 Průměrné časy a FPS při použití 1 kamery 1280×720 36

7.6 Průměrné časy a FPS při použití 2 kamer s rozlišením 1280×720 . . . 36

7.7 Průměrné časy a FPS při použití 2 kamer s rozlišením 1280×720 a 1 s rozlišením 640×480 36

7.8 Chyba v závislosti na vzdálenost při použití jedné a dvou kamer. Hodnoty jsou v centimetrech. 38

Přílohy

A Obsah přiloženého CD

- `LokSys`
  - `input` ..... příklady vstupních souborů
  - `src` ..... zdrojové kódy implementace
  - `tools`
    - `ez_calib.py` ..... nástroj na kalibraci vnitřních parametrů kamery
    - `simple_client.py` ..... příklad jednoduchého klienta
  - `README.md`
  - `CMakeLists.txt`
- `text`
  - `bakalarska_prace.pdf` ..... tento dokument ve formátu PDF